The Amateur Computerist

Webpage: http://www.ais.org/~jrh/acn/

Winter/Spring 1992 **Volume 4 No. 2-3**

"Any society that is alive is a society with a history."

Vaclav Havel

WELCOME TO THE WONDERFUL WORLD OF COMPUTERS vs PLANT CLOSURES GM STYLE

4 Years of Amateur Computerist

(Editor's Note: With this issue the *Amateur Computerist* begins its 5th year of publication. Also, we are including in this issue, a complete index of back issues.)

February 11, 1988 was the first issue of the *Amateur Computerist*. The issue was "dedicated to the Flint Sit Down pioneers on the victory of their battle to win industrial unionism 51 years ago." ("Dedication," vol 1, no 1)

In our first issue, we wrote: "There was an effort by administrators of the UAW-Ford program at the Dearborn Engine Plant to kill interest in computers and computer programming. We want to keep interest alive because computers are the future." ("Introduction," vol 1, no 1)

One UAW pioneer, Jack Palmer, in summing up the heritage the sitdowners were passing on, wrote:

"Each generation has to solve its own problems. The sit-down

generation solved the problem of organization. The postwar (*WWII -ed*) generation solved the problem of pensions and inflation. Not entirely, but a good start was begun. The present generation is faced with the greatest problems of all.

Table of Contents

They are Automation, Peace and Politics." (from the *Searchlight*, newspaper of UAW Local 659, Flint, MI, April 21, 1960, p. 2, quoted in "Dedication," vol. 1, no. 1). The *Amateur Computerist* is an effort to carry forward the torch passed on by the sitdowners – the need to solve the problem of automation.

The welcoming of this newsletter by Floyd Hoke-Miller, a pioneer of the Flint Sitdown Strike, demonstrated that there was a continuum between the Sitdown generation and the current generation. In his article, "Dawn of A New Era," Floyd wrote, "From the Great Wall to the Great Pyramid, from the hieroglyphics to the screen of the computer, mankind is still progressing. So make the new born science, that has given us the computer for the amateur and not as a prerogative of the professional to be shrouded in secrecy from humanity, the choice of the individual, not an election of a minority. (vol 1, No. 1)

But what has happened in the past 4 years? Has automation in the

auto industry made any headway? Has the use of computers and computer education gone forward?

In February, 1992, GM announced plans for massive layoffs and plant closings in North America. The high price of automobiles in the U.S. shows that the problem of automation in the auto industry has not been solved. Instead there have been disincentives to the introduction of new technology in the U. S. auto industry similar to those that led to the unraveling of the Soviet Union's so called "command economy." A recent article in *Fortune* Magazine (vol. 125, no. 1 January 13, 1992 "Can GM Remodel Itself?" by Alex Taylor III) describes GM's inability to utilize new technology. "For example," Taylor writes, "GM put $77 billion into new plants and equipment to reduce labor costs.... Some robots it acquired in the mid-1980s stand unused today. The highly automated equipment never delivered the promised savings because GM did not train workers properly to use it, and...failed to design new models for easy robot assembly."(p. 34)

As articles in previous issues of the *Amateur Computerist* have shown, investment in the U.S. auto industry in the 1980's was not in automation that would result in less labor being needed to produce an automobile. No effort was made to reduce the price of a car so more people could afford to buy cars.

Instead, U.S. autos were downsized and redesigned, and some of auto company profits were used to set up so called "joint" labor-management labor relations programs or to invest in other companies or abroad. (See for example, "Labor Relations Hoax," vol 2, no. 4) U.S. auto companies instituted a plethora of cost cutting initiatives which removed the incentive for investment in new technology. For a company to invest in modern technology, labor saving devices must cost less than the labor they are replacing. (See "Shorter Hours Are Needed....", vol 2, no. 3)

Record profits had been reported in the industry in the 1980s. Yet Wall Street analysts like Maryann Keller sharply criticized new technology being introduced into companies like GM. Instead GM adopted the model of Japanese labor relations introduced into Toyota in the 1950s. This model had been developed based on research that GM conducted during a 1947 labor relations experiment which they called

the "My Jobs Contest." (See "When Will Their Walls Come Tumbling Down," vol 3 no. 1) Once again in the 1980s and now in the 1990s such labor relations schemes are being heralded as the savior of a "failing General Motors" and as a model for the rest of U.S. industry. For example, the design for GM's new Saturn plant was premised on GM's commitment to making labor more intense rather than introducing labor saving technology. A spokesperson explains Saturn's philosophy:

"In Saturn, we believe that to automate...does not make good business sense. When you consider the capital investment required for robots or other automated systems, you have to look at the variable labor cost as an alternative. Automated equipment is limited! On the other hand, people, if properly trained, can and have taken costs out of an operation when given the opportunity." (Joseph F. Malotke, *Labor Law Journal*, Aug. 1985, p. 568)

GM's Saturn model is a reversal of lessons learned over the past 50 years about the social benefits of new machinery. As Howard Foster, a UAW pioneer in Flint, describing the benefits of modern machinery, explained: "When mass production methods were introduced in the automobile industry, the price of cars went down. This was because the labor time on each car was greatly reduced. Yet we auto workers got higher wages through our union." (See "Technology: To Develop or Stagnate," vol 1, no 2)

In opposition to similar labor relations experiments, GM auto workers in Flint, MI, in 1947-48, successfully campaigned for wage increases that would match inflation. Their battle against GM management and the UAW International Union officialdom who opposed such wage increases, was victorious. Thus pattern setting wage increases known as COLA (Cost of Living Adjustment) and AIF (Annual Improvement Factor) were introduced into the UAW-GM contract in 1948. These wage increases made it profitable for GM to introduce new machinery and update production methods in GM plants.(See "Technology, to Develop or Stagnate," vol 1, no 2)

By 1950, contract language describing the AIF affirmed the social desirability of using new technology to decrease the labor needed to produce an auto. The 1950 GM-UAW contract granted workers a wage increase based "on technological progress, better tools, methods,

processes and equipment and a cooperative attitude on the part of all parties in such progress. It further recognizes the principle that to produce more with the same amount of human effort is a sound and economic and social objective." (See "Upcoming Elections and Computers," vol. 1, no.3)

Union-management jointness experiments of the 1980s are a violation of the principle "that to produce more is a sound and economical objective." The UAW-Ford National Development and Training Center founded in 1982, and similar experiments at GM and Chrysler, are labor relations experiments. Their aim is to "promote training, retraining and development activities," instead of "technological progress, better tools, processes and equipment," (See "Labor Relations Hoax," vol. 2, no. 4)

Because their aim is not education in high technology, there was a battle at the Dearborn Engine Plant over the cancellation of computer programming classes. UAW members wrote: "How can UAW members be trained in high technology by cutting computer classes out...we sent letters everywhere. We are tired of being denied benefits we're entitled to. We are tired of being shuffled from one person to another to cover up who we're fighting...we can't sit back and let happen at the Rouge what has happened at GM – the wholesale closing of plants...WE NEED AN INVESTIGATION INTO WHAT IS GOING ON IN THE UAW FORD PROGRAM at the Dearborn Engine Plant." (See "When Will Their Walls Come Down," vol. 3, no. 1)

In a subsequent issue of the *Amateur Computerist*, Floyd Hoke-Miller contributed articles explaining why there is more thinking needed among those who do the work of the society. "We, as the working class," he wrote, "are the 'low men on the totem pole' or like Atlas 'bearing the world on our shoulders'." (See "Pass the Profits, Please," vol. 1, no. 2) He championed the need for fewer hours of work to have any benefit from new technology and reminded workers that they are still working at least the same 8 hour day "gained by a hard struggle of the transportation unions prior to WWI." (Ibid., See also "Shorter Hours are Needed for Computers to Benefit Labor", vol 2, no. 3)

The problem of the 1990s is similar to the problem of the early post WWII years – how to utilize significant new technological developments

to make life better for the people of the society. Large scale industry needs to be under some form of social control and strict regulation for technological progress to serve the society.

The birth of the personal computer in 1974 and its development and evolution into the 1990s has put the promise of a better world on the horizon for people in the U.S. and around the world. At the computer classes at the Ford Rouge Plant, before the programming classes were cancelled to make way for the labor relations experiments that replaced them, one auto worker wrote: "Welcome to the Wonderful World of the Computer." The cancelling of those classes in 1987 has made clear that that wonderful world cannot be gained by going backward to ever more labor intensive production, (i.e. speed up), less and less manufacturing capacity (shortages), and ever increasing prices. This is the economic program that U.S. corporations and the U.S. government are promoting in Eastern Europe and the result is massive human hardship and economic dislocation. This economic program will result in the same economic dislocations in the U.S. economy as are currently plaguing Eastern Europe.

Thus once again, as in the immediate post WWII period, there are serious economic problems in the U.S. and the world. There is a need to have auto workers and others, particularly those who work in large scale industry, consider and debate the economic problems of our times. History has shown that this is the only way that solutions which lead the society forward can be found. Prohibitions against workers writing in their local union newspapers to prevent public criticisms of trade union officials or policies, as exist in the UAW, need to be overturned. (See for example, Convention Proceedings of 1951 UAW Convention and UAW Public Review decisions like No. 888, Bier vs Local Union 2500 Executive Board, UAW, and No. 238, Plyer vs Local 599 (1961))

As Carl Johnson, a UAW pioneer explained, auto workers are capable of solving the problems facing them if they have access to organs of free discussion. Johnson wrote: "If local union publications...provide the ranks with a freer discussion which alone can prepare the ranks for the fight which is sure to be plenty tough, then we need not worry too much, for American labor proved in '36 and '37 that it can move fast and furiously when it knows where to go." (Carl Johnson,

"Only More Democracy Can Save Democracy," Feb. 1, 1945, *The Searchlight*, UAW Local 659, Flint)

Describing the important role to be filled by an uncensored labor press, he wrote, "If the Labor press does not try to give Labor the whole truth, where will Labor get it? This, of course, raises the question: Who is right about Labor's destiny? Certainly we can't rely on the capitalist press to tell us, for it is obvious that their interest is the opposite of Labor's interest. But who, from the ranks of Labor? Let them all speak – that's what Free Speech was intended for! Let them all present their view in a forum. From that the reader will have a fair chance to decide." (Ibid., Oct. 29, 1949)

Once again, in 1992, there is the need for access to the uncensored local union newspaper. This tradition was pioneered by auto workers in Flint in the 1940's to carry on the militant Spirit of the Sit Down Strike, the "Spirit of '37." This press made it possible for those who do the work of the society to be able to analyze the current problems and debate the way forward. This uncensored labor press no longer exists, but computer bulletin boards across the U.S. and the world are providing access to uncensored discussion so the momentous problems of our times can be debated and analyzed. On computer bulletin boards like Usenet News, MNET (Ann Arbor, MI), etc., such discussion is now taking place. (*We plan to include articles describing this important development in future issues*.) Also, there is a need for education, independent of large corporate employers like GM or Ford, and for a press to express a working class voice, independent, as well, of any censorship from union officials. To this end the *Amateur Computerist* is dedicated and we need and welcome your participation in helping to create this independent voice so as to be able to gain the fruits of the computer revolution for those who do the work of this society.

Floyd Hoke-Miller, a pioneer of the Flint Sitdown Strike wrote the following poem as part of his commitment to the need for an uncensored press:

Voice of the Chevrolet Worker

by Floyd Hoke-Miller

It matters not what bossman say,
How much they rant and rave
Their Sunday suit and higher pay,
Do not exclude the grave.

***

The wage-slaves toil at their behest,
Producing only by their word
There's no denying one request,
Their voices must be heard.

***

They know quite well that banker men,
And owners of the tools
Connive with pie cards when they can,
To treat the laborers as fools.

***

Their language may not stand all tests,
But let them have their say
For on their backs the burden rests,
They MAKE the Chevrolet.

AMATEUR COMPUTERIST INDEX (5 Year)

Volume 1 No 1 Feb. 1988

WHY LEARN PROGRAMMING
COVER OF PERSONAL COMPUTING (PICTURE)
COMMODORE TIPS & TRICKS

Volume 1 No 2 Jun. 1988

THE BIG MACHINE
PASS THE PROFITS, PLEASE
TECHNOLOGY: TO DEVELOP OR STAGNATE?
CARTOON BY “DOC” WILSON
SAMPLE BASIC GRAPHIC PROGRAM
TRY THIS (IBM)
THE WORLD OF TELECOM... CORRECTIONS
GERMAN VOCABULARY HELPER
PROGRAMMING IN BASIC OR C?
CONFIGURING YOUR SYSTEM
LETTER TO THE EDITOR

Volume 1 No 3 Oct 1988

LETTER PUBLISHED IN RADIO-ELECTRONICS
RESPONSES FROM AROUND THE COUNTRY
EDITORIAL: SAVIOR IN WAITING
HOW TO USE THE MERIT NETWORK?
VIRTUAL DRIVES & BATCH FILES
TRY THIS (EQUATION OF A STRAIGHT LINE)
AS I WAS SAYING... (WHY COMPUTERISM?)
COMPUTERS AND FREE SPEECH
LETTER TO THE EDITOR
PLANT LIFE (PICTURE)

Volume 2 No 1 Jan. 1989

RETURN TO SANITY WITH THE AMATEUR & THE PRO
LETTERS FROM READERS
PROBLEM CORNER
TRY THIS FOR IBM (INPUT NUMBER FROM 20-150)
SYSTEM DIAGRAM FOR QUADRAPHONIC SOUND SYS.)
RESPONSE TO OCTOBER EDITORIAL
CARTOONS (COMMODORE COUNTY)
WELCOME TO COMMODORE COUNTY USA
COMPUTER HACKING, A CRIME?
IBM KEY ASSIGNMENTS USING THE “PROMPT”...
HISTORY OF COMPUTERS... PART I

Volume 2 No 2 Apr. 1989

WHY LEARN TO PROGRAM? (DISCUSSION)
LETTERS
TRY THIS (MESSAGE)
“SE Q” FOR IBM
AS I WAS SAYING (JOBS: HOURS AND SENSE...)
OVERTIME AND UNDER PAY
MAY DAY
SAMPLE BATCH FILE
HISTORY OF COMPUTERS PART II

Volume 2 No 3 Summer 1989

IMPACT OF COMPUTERS ON SOCIETY: A DEBATE
LETTERS TO THE EDITOR
COCO CORNER (GRAIL QUEST-PIP)
COMMODORE COUNTY USA (CURSOR COLOR CHANGE)
OUT OF THE HEART OF THE ABACUS...
HISTORY OF THE COMPUTER PART III

Volume 2 No 4 Fall 1989

LETTER FROM PROSECUTOR
OPPOSING VIEWPOINT...
LETTERS TO THE EDITOR
WANTED ALIVE (AD)
COCO CORNER (EQUATION GRAPHING PRG.)
TRUE HEROES
TRIGONOMETRY LESSON FOR IBM
HISTORY OF THE COMPUTER PART IV

Volume 3 No 1 Winter 1989

LETTER FROM EDITOR OF DETROIT NEWS
DON’T REPLICATE UAW-FORD SCHOOL
LETTERS TO EDITOR
COMMODORE COUNTY USA (CARTOON)
THE SPIRIT OF BABBAGE
COCO CORNER (POKE & PEAK)
CAD/CAM/CIM
HISTORY OF COMPUTERS PART V

Volume 3 No 2 Spring 1990

THE LABORER, YES
FLOYD HOKE-MILLER (1898-1990)
THE PICKET

IN HONOR OF LABOR'S POET LAUREATE
COMPUTER EDUCATION AND GOVERNMENT REG.
LETTER FROM SUPERINTENDENT
OPEN LETTER TO SUPERINTENDENT BEMIS
LETTER TO GOVERNOR
COMMODORE COUNTY U.S.A. (SHIMMERING TEXT)
C64 MUSIC DIGITIZER
IBM LABEL PROGRAM
COCO CORNER (CALORIE COUNTER)
BULLETIN BOARD NUMBERS

Volume 3 No 3 Fall 1990

WHAT CRITICISMS HAVE YOU OF THE A.C.?
TIPS AND TRICKS (IBM BOOT PROBLEM)
LETTER TO EDITOR
EDITORIAL
A COMMON MAN OF GREATNESS
COCO CORNER (CORRECTION)
EXCERPTS FORM BBS (DISCUSSION-TRADE UNIONS)
COMMODORE C64 RESET SWITCH
DIAGRAM #1 (FOR RESET SWITCH)

Volume 3 No 4 Winter 1990

HATS OFF TO PATRIOT
AMATEURS ARE NEEDED MORE THAN EVER
COCO CORNER (MORE POKE & PEAK)
BRINGING AUTOMATION HOME
COMPUTER BBS DISCUSSION ON THE WAR
COMPUTERS FOR THE PEOPLE: PART I

Volume 4 No 1 Fall 1991

COMPUTERS FOR THE PEOPLE - A HISTORY PART II
LETTERS TO THE EDITOR
TEN COMMANDMENTS OF GOOD NETWORKING
TRY THIS PROGRAM (GRAPHIC "HI")
THE USSR AND THE COMPUTER
COMMAND LINE CALCULATOR
THE QUESTION OF CENSORSHIP

PROBLEM CORNER

I have a problem and I'm hoping that somebody can help. I upgraded my XT compatible by installing high density floppy drives and they work just fine, but when I put a 360K or a 720K diskette in, it will not read it unless I reboot. After using the low density diskettes I must reboot again before I can use high density diskettes. Is there something I can do to make the computer read the floppy whatever the diskette is?

PUZZLED

UNION FOREVER

by Floyd Hoke-Miller

Tho' politicians come and politicians go
Let Unionism go on forever.
But vote for him whose records show
He kept the workers' cause his first endeavor.

Now has he fought for higher wages
And the thirty-hour week;
The dream through the ages
Of the lowly and the weak.

Because machines are taking powers
That were our jobs of yesterday
But it's the same old tedious hours
With the same old lousy pay.

Letter To The Editor

(Editor's Note: The following letter by a staff member of this newsletter was recently published in the *Columbia Daily Spectator* in N.Y.C.)

To the Editor,

Probably unbeknownst to many students, the rally and entering of Low Memorial Library on Tuesday, February 11, 1992, was in the tradition of a glorious victory that took place 55 years ago. On February

11, 1937, General Motors auto workers in Flint, Michigan emerged from the factories they had occupied, victorious. As part of a long series of strikes nationwide, this forty-four day long Sit-Down Strike won GM auto workers the right to have the United Auto Workers (UAW) represent them as their bargaining agent to GM. This was the beginning of the official recognition of the UAW by auto manufacturers.

The sit-in at Columbia to achieve more of a student voice in University decision making is a poignant reminder of the similar fight that occurred 55 years ago in Flint, Michigan. Just as those workers fought the auto industry for a voice, the students of Columbia are likewise fighting to have their voice heard. By learning more of the tradition behind the battle for democratic rights, we shall be stronger.

Michael Hauben

Letters to *Amateur Computerist*

Dear Editor:

Along with your history of computers, there is one bit of government action your readers might find interesting. In the late ‘50s the U.S. government had at least three competitors in the running for a contract to design a computer utility. The phone, the electric, the gas, and the computer were all to be durable and available for Americans.

The names that took part in the design competition were Dartmouth with its DTSS time-sharing system, Bell Labs with the forerunner of the now popular Unix, and MIT with Multics.

Unix, you are all familiar with now. Dartmouth time-sharing, last I knew, was still a basic entry level operating system running at least at GM Tech Center on a dual big box Honeywell system.

MIT won the contract with its Multics operating system. General Electric won the contract to build the iron to fit the then existing software. The GE645 computer hit the field in 1962 with absolutely no fanfare whatever. (The government’s support evaporated somewhere around this time.) Multics languished around the military and institu-

tions for the next twenty years. Honeywell entered the picture when the government refused to allow GE to buy out Honeywell's computer department. So instead they sold out to Honeywell. Honeywell Info Systems was a poor company. The development money was non-existent. With the financial crisis of the '70s many attempts were made to terminate Multics. The necessary iron to run the system was more complicated than Honeywell was willing to spend the money to speed up, so the generation of vlsi was never developed. Honeywell Info Systems was in big trouble in the '80s, the French bought the company and it is now _Bull.

As to software, Multics is still the undisputed king of security, flexibility, purity of design, and ease of use (after learning). Near what appeared to me to be the end of its troubled history, Multics had earned the B2 level of security from the National Security Administration. If anyone listened to them, Multics is the only machine any government entity is allowed to use. The main problem with selling or describing Multics is that no outsider has any frame of reference to it. Ford Research Center is the largest user of Multics. Management made a determined effort to stop development and get off Multics when the product was scraped by Honeywell in 1985. To this day no replacement has been found to even approximate the most basic functionality inherent in Multics from its early days.

The system is fully paged and segmented, gated, ring bracketed protected and written 99.96% in PL/1. Even most of the PL/1 compiler is written in an escalating style of PL/1. Most big box operating systems have a known errors file a foot thick. Multics had no known errors. Ford did run into an upper limit in that no single array could be larger than 1 megabyte and you could have no more than 256K segments (programs, objects) open at one time to a single process (the program in control of the job flow). Locally they moved those limits up a ways.

When I left, two systems were in use by Ford world wide and were as large as the iron would allow. Six processors, 64 megabytes of memory, a sea of disk drives and every known style of communications device and protocol. A system, new in 1985, had cast the ultimate computer for the world, that of, Multics hooked at 50 megabaud to a Cray XMP.

Rick Strome
Dearborn
Multics hardware repair till 1986

Dear Sir,

I have read in *Hacktic* that you make the magazine *THE AMATEUR COMPUTERIST*. I don't know the price of your magazine or what the magazine contains therefore I have a question. What is the price of your magazine and how can I be a member? I hope you would answer my questions and I would appreciate it when you send me a sample issue. I am looking forward to receiving your reply, meanwhile hearty thanks.

Yours Truly
R. H. SMIT
THE NETHERLANDS

Open Letter to Editor of *Utne Reader*

(Editor's Note: This letter was sent to the editor of the *Utne Reader*. It was not published.)

Dear Editor,

Instead of pointing to important articles in the 'zine press in "Notes from Underground" (Nov/Dec issue), your article mis-characterizes this section of the alternative press as 'unconventional' and 'obscure.' For example, you write:

"There are few 'zines coming from minority or working-class communities... proportionally fewer women publish 'zines than men, and when they do, they tend to be feminist or Pagan-oriented."

I am one of the editors of a 'zine that comes from a working class community and this 'zine is neither 'feminist' nor 'Pagan-oriented.' A recently published book, *Technoculture*, by Andrew Ross and Constance

Pawley, (University of Minnesota Press, 1991, p. 125) describes our newsletter:

"When worker education classes in computer programming were discontinued by management at the Ford Rouge Plant in Dearborn, Michigan, United Auto Workers members began to publish a newsletter called the *Amateur Computerist* to fill the gap. Among the columnists and correspondents in the magazine have been veterans of the Flint sit-down strikes who see a clear historical continuity between the problem of labor organization in the thirties and the problem of automation and deskilling today. Workers' computer literacy is seen as essential not only to the demystification of the computer and the reskilling of workers, but also to labor's capacity to intervene in decisions about new technologies that might result in shorter hours and thus in 'work efficiency' rather than worker efficiency."

It would be good to have *Utne Reader* include worthy reprints from 'zines like the *Amateur Computerist* as this is a part of the alternative press which provides a rarely heard voice in American society.

Sincerely, Ronda Hauben

Review from the *µPERIPHERAL* ISSUE 20 FEBRUARY 1992

(Editor's Note: This review of the *Amateur Computerist* appeared in an Australian computer newsletter. We are reprinting it because of the interesting historical coincidence it points out regarding the appearance of the first Australian kit computer distributed in a popular electronics magazine.)

THE AMATEUR COMPUTERIST

We received an exchange copy of Vol 4, No 1, for Fall 1991, from Ronda Hauben, P.O. Box 4344, Dearborn, Mi. 48126 USA. A twelve page double column issue, featuring how hackers gave birth to the

personal computer (second part of a four part series) covering ENIAC, David Ahl and *Creative Computing*, the Dartmouth Basic of Kemeny and Kurtz, Ted Nelson's wonderful book *Computer Lib*, and Jonathan Titus' Mark 8 computer in *Radio Electronics*. Incidently, Jim Rowe's all TTL EDUC-8 in *Electronics Australia* missed out by one month in being the first kit computer circuit distributed in a popular electronics magazine, and it was the Mark 8 which appeared first.

OK, other articles include ten commandments on networking (mostly Novell on PCs), a short on USSR computers, a command line calculator in Quick C which does trig, powers and root (about 300 lines) plus letters.

Ronda asked by e-mail whether anyone knows of other computer magazine editors who can be reached by e-mail? Her address is:

au329@cleveland.freenet.edu

(*µPeripheral* is available from Eric Lindsay, 6 Hillcrest Avenue, Faulconbridge, NSW, 2776, Australia. Electronic Mail Address: eric@zen.maths.uts.edu.au.)

Tribute to a Modern Computer Pioneer: Grace Hopper (1906-1992)

A *New York Times* headline on January 3, 1992 read: "Rear Adm. Grace M. Hopper Dies; Innovator in Computers was 85." She had died at home New Year's Day after a recent illness. Until then, Grace Hopper had been involved with computers and computer programming since 1944. As one of the pioneers of modern computing, her life exemplifies the activities and goals that propelled computing forward.

Hopper joined the U.S. Navy during World War II. Having a Ph.D. in mathematics, after Midshipman's school, she was assigned to the Bureau of Ships Computational Project at Harvard University. Her first day at the Project, she met Howard Aiken and his new calculating apparatus. The Mark I, as it was called, was 51 feet long, 8 feet high, 2

feet thick, weighing 4 tons. It had three quarters of a million parts, 500 miles of wiring and three million wire connectors. In order to operate this first operational program controlled computer made in America, rotary knob-like switches had to be set manually. Hopper was fascinated by the challenge of such a gadget having been a tinkerer when she was a kid. One story is told of how as a girl she dismantled and reassembled all the family's alarm clocks. Aiken handed her a code book and asked her to work out the coefficients for the interpolation of the arc tangent function to be entered into the Mark I. It was a sudden but exciting introduction to her first computer. She was 37 years old at the time.

Aiken helped all his colleagues deal with their new tasks by suggesting that they read portions of Charles Babbage's writings. Hopper appreciated the importance of such reading. She recalled that when she was growing up, "Each summer we had to read 20 books and write reports on them. You were educated and had some background when you were through then. It gave us an interest in reading and in history." Aiken also assigned Hopper to write the first operation manual for the Mark I which she did as well as eventually writing over 50 articles, especially on programming.

The Mark I was the first computer to be sequentially programmed and Hopper was at it from its beginning. Since at the time programming essentially meant writing out strings of numbers as switch settings, coding, operating and plugging instructions, etc., errors could easily be made. Also many strings of numbers had to be repeated frequently. So the habit arose of writing out pieces of code that were already checked out in notebooks and passing the notebooks around to be copied from when needed.

During the summer of 1947 there was trouble with the Mark II computer (successor to Mark I.) The trouble was traced to a mechanical relay in which a moth had been trapped and beaten to death. The body of the moth was removed with tweezers and taped into the log book as the cause of the problem. Grace Hopper is given credit for coining the term "bug" for computer problems and for the explanation to Aiken when he asked what's holding up the numbers, that she and others were "debugging" the machine.

Hopper left Harvard in 1949 and joined the Eckert-Mauchly

Corporation which was working on building its UNIVAC I. Programming for computers like Mark I, II, III and UNIVAC I was necessarily in full detail including at times the specification of individual bit patterns and number strings. But many programs contained identical subroutines or sub-programs even though the total objective of such programs may have differed. Hopper energetically encouraged the gathering of such subroutines into permanent subroutine libraries. She also spearheaded an effort to program computers to utilize such subroutines. The idea was to create a program that could receive as input a set of high level spoken language-like commands and produce as output an integrated program made up of appropriate subroutines. In order to create such a program, Hopper had to overcome the prevailing prejudice that computers could do wonders at arithmetic but could not do analytic work like programming. She and her team at Rand Corporation succeeded at creating the A-O and other compilers that showed the cynics were too limited in their expectations of what computers would be able to do.

Also in her quest to show that the new machines were more than number crunchers, Hopper sought to demonstrate their analytic capabilities. One such program she wrote was designed to take as input mathematical functions and gave as output the derivative of such functions. Upon seeing the program perform, one researcher who had spent months finding the first 15 derivatives of a complicated function, insisted that Hopper must have had some hidden person feeding the derivatives into the computer. He felt no machine could do in eighteen minutes what it had taken him six months to do. (from Robert Slater, *Portraits in Silicon*, Cambridge Mass, 1987)

Hopper had succeeded in demonstrating that the computer was basically a symbol manipulator: whether the symbols were numbers, letters, words or other data structures, was a detail for the programmer, not a complication for the computer. She drove this home by writing a compiler that could receive high level code written in English, French or German. Again, someone who saw this program in operation could not believe a computer made in the U.S. could understand European languages and wondered what the trick was. Based on this work, by 1957, Hopper and her staff had created “Flow-matic,” the first computer

language employing words. (Ibid., p. 225)

Hopper's work on "Flow-matic" was seminal and was followed by other achievements such as Commercial Translator by IBM. But Hopper and others saw a danger in the prospect of having many different languages. She was, therefore, part of the process of creating COBOL, an easy to read machine-independent language not identified with any specific computer manufacturer. There were those who denounced COBOL because of its flaws and a rumor spread that it was dead. But COBOL has in fact, like BASIC, opened the door to a significantly large universe of users and it is still in use today 30 years after its introduction. Her work in support of COBOL was part of Hopper's campaign for standards for languages, architectures, data structures and networks, standards set for the use of all and not by any dominant firm.

From 1944 until her death New Year's Day 1992, Grace Hopper spent her life pushing the use of computers forward and also spreading that use. She especially enjoyed the opportunity to challenge young people to explore and develop the most infinite possibilities she saw inherent in computer technology. Her understanding was that we are today just "at the very beginning of the mass use of the computer. We haven't even begun to exploit its potential." (Marguerite Zientara, *The History of Computing, A Biographical Portrait of the Visionaries Who Shaped the Destiny of the Computer Industry*, Framingham, 1981, p. 53)

Interview with Staff Member Michael Hauben on the Occasion of the 10th Anniversary of the Personal Computer

Part I

(Editor's Note: This interview was conducted on August 11, 1991. It has been edited.)

Ronda: Tomorrow is the 10th anniversary of the introduction of the

IBM personal computer on August 12, 1981. Also, one of our staff members, Michael Hauben, is leaving Michigan to go to college in N.Y. Therefore, it seemed an appropriate time to look back on the past 10 years and to review how the introduction of the personal computer has affected our lives. Michael is now 18. In 1981 he was 8 years old and already involved with computers. Michael is not only one of the beneficiaries of the computer revolution. The computer revolution was carried out, not so much by companies like IBM, but more importantly, by computer hobbyists like Michael Hauben. Thus in honor of the computer hobbyists, who gave birth to and developed the personal computer, we would like to review some of your experiences, Michael, with the computer.

Bill: How did you get started with computers?

Michael: The first place I really saw computers was at an exhibit in Toronto over 10 years ago. There was a robot that was like the 4 axes machine that auto workers use. They also had a computer exhibit. I don't remember what kind of computer was on display but they were just a bunch of computers running different kinds of programs set up there at the Canadian National Exhibit. That really peaked my interest somehow.

When I was 8 (in 1981), I took a computer class at Schoolcraft Community College, in what was called the Kids College. It was part of what they called the TAG (Talented and Gifted) Program. The teacher's name was Mrs. Brown. We learned on the Apple II+'s. The first day of class, Mrs. Brown lifted the top of the APPLE and said, "There, that's all there is to it, There's nothing to be afraid of." That was a very good introduction to the computer because it showed there was nothing to be afraid of. That we could completely control it. I learned BASIC there. I took several other classes in that program. I think I took three. I didn't take all the BASIC language classes offered. But I took a test that they had for their normal BASIC college level classes and I wound up getting three college credits for the BASIC language class. And I didn't do so good because I ended up only getting a B on the test. But the experience was interesting and from then on whenever there was a computer available I tried to use it.

After the trip to Toronto, I always wanted to buy a computer. There was the Texas Instruments 99/4a (TI 99/4a) and I don't remember how

much it cost, but it was expensive. There was the Timex Sinclair 1000 (TS 1000) and that was much cheaper. My family and I had seen Sinclair computers in England when we visited. These computers could be hooked up to a normal TV set. I saved up my money and bought a TS-1000. Using it I more thoroughly learned BASIC. My father and I programmed a lot in BASIC with only 2K memory. We never seemed to run out of memory. We just played around and tried to do lots of different things, tried writing little games, graphics and we dabbled a little in machine language, not a lot however. Whenever I had the chance, whether it was summer camp or in a computer store, I'd try to do something with the computer. I learned BASIC, I learned LOGO on the TI-99/4a in Camp, and I played around with APPLES and with Commodore PETS. In my elementary school, there was a terminal hooked in with the mainframe of the Dearborn Schools. At that time there were many programs on the mainframe. They had BASIC. They had games like the OREGON TRAIL, etc. I subscribed to two or three magazines for the TS-1000. I bought books, did all the TRY THIS type of small programs. Those were always fun because there would always be problems with the programs. There would always be 'bugs.' The books and sample programs were exciting somehow. I haven't found many books similar for programming on the IBM PCs today, books that I have found exciting for a hobbyist. And this is sad.

Soon after I bought the TS-1000, it couldn't have been more than a couple of years, I was trying to choose between the TS-2068 and the Commodore 64. I think the Commodore was more expensive. The TS-2068 had better color, and a more developed version of BASIC. The Commodore 64 was better in that it had a disk drive and the TS-1000 only had a tape drive you could use. The Commodore also had a real keyboard, while the Timex utilized raised chicklets. I bought the TS-2068. Then I had my first real lesson in the computer world. Three months after I bought the TS-2068, Timex stopped selling it and supporting it. Timex made a deal with Commodore. There was an agreement to sell the Sinclair in England and Europe and Commodore in the United States. That was a shock because I thought I made a better choice, but it turned out the better deal is not always the best choice.

And my father and I did programming on that, but not really as

much as we did on the TS-1000. It was a lot less, even though there was the added attraction of the color and the sound and the joystick port. And so I still did things and I tried to pick up on things whenever I could.

Christmas of 1984, we bought a Sanyo MBC-550-2 which was a MS-DOS compatible, but not an IBM compatible, machine. The operating system was IBM compatible, but the graphics were different, the sound was different, and the BASIC was different. The Sanyo was a better machine for graphics, I think 640 x 400 with 4 colors if not 16. And WordStar worked. That's why my family got it – as a wordprocessor. I learned MS-DOS. I got more into the PC world. We subscribed to a Sanyo magazine for a while. We went to the Sanyo Users' Group for a while. We occasionally went to SEMCO (Southeast Michigan Computer Organization), but somehow that was already oriented toward business and they weren't very interested in helping us. Then in 1985, through INACOMP, my mother won a Compaq Portable. It was one of the earliest to come out that was fully IBM compatible. It was a luggable portable, and it weighed about 20 pounds, if not more. And that's how I really got into IBM. We had a choice between a modem and a hard drive. We got a modem. It was a breakthrough. The hard drive seemed important but the modem was more important. We wound up getting a hard drive later on. With the modem, it lets you connect to the outside world. With your own little system you'd be like a hermit, but in connecting with the rest of the world, it's other people's opinions, different discussions about computers, about current events, debates about what's going on in the world and just general BS also. And you came into contact with people, you came into contact with different files to use with your computer, with what was going on with the computer scene and so somehow it was like a replacement for a user group. And depending upon the time, there was either a lot going on or a little going on.

Ronda: What do you mean?

Michael: Well right now not many boards I know have much debate on them. There are two that I am on. Both of them have debates on-going. I'm sure there are others, but I just haven't had time to look. But for a while I was on many of the boards and at one point many of the boards were silly contests to see who could post the most numerous

messages.

Ronda: Do you have a sense what you were looking for on the BBS's? You used to spend a lot of time on them.

Michael: Well at first I wasn't on local BBS's. Originally, I was on CompuServe.

Bill: Free time?

Michael: Well, the first two hours were free. I almost became a Beta Tester for InfoCom through CompuServe. I sent in the application forms. I then received a congratulations letter, but InfoCom never sent me any games to test. The only response was a Christmas card. That was a soured CompuServe memory. I found some local BBS numbers listed on CompuServe and from my father and some friends of his from work. For a while I was mostly on Commodore BBS's and not many IBM boards. But then I started calling the IBM boards. It was new for me when I started. Modeming was a connection to the outside world to other people with similar interests. It was interesting – the debates about current events. Somehow there was the possibility for intellectual discussion which I couldn't find elsewhere besides my parents and a few friends like Floyd Hoke-Miller. But among my friends at school or neighbors, there wasn't much of a possibility.

When we lived in East Dearborn, our next door neighbor, Tom, had an Atari and a Commodore 64. He shared an interest in computers with me. He was my friend, even though there was a large age gap, because we were both interested in computers. He let me come over and try some things on his computer and I'd go with him to computer stores.

Bill: Another thing about modems you can't tell the age. Treats you more like an equal.

Michael: There's an anonymity. You don't know anything about the other users. So you are more willing to accept them. There are still first impressions. If you act like a real idiot, people won't like you. But the full element of first impressions is left out. And people tend to rank you or be friends with you on how you act online, what you speak about. It does help. You tend to get to know the people and there isn't as much blocking. And my first handle was Wizkid. I changed my handle 2 or 3 years ago to Sentinel. And there was one person who signed on and said it was great knowing you. He was one of the people who knew me as

Wizkid. There was a "Remembering the OLD Days" theme area on one of the BBS's and someone said, "remember that Wizkid." And I said, "that was me." And he said he didn't know that. When people change their handles, it's public but somehow people don't always realize it. When I changed my handle, I decreased my activity. When I decreased my activity it was because there were just silly messages that didn't mean anything, or they just seemed juvenile, and I don't know if that's because the people calling were younger or they were more juvenile. The way people accept you is based on your maturity online and your maturity showed through more than your age. And there was one debate where someone said you are just a kid. And I used to have the handle Wizkid. But it didn't matter what your age was, it was more how mature you were. He was trying to say "Well you're just a kid, you can't know anything." But he was wrong. So there is less age discrimination on the boards.

Ronda: Why did you decrease the time you spent on the boards?

Michael: I had to spend more time with school, with friends, with my job. Whenever I used to come home from school, I used to spend two or three hours, but then my mom said, "We need the phone." So I didn't spend my free time before homework on the modem. And then with work, I wasn't even home on certain days to use the modem.

Ronda: But it seemed you were also a little disappointed. There were user parties, but it seemed the computer world didn't extend outside of the modem.

Michael: It did to a certain extent, but it didn't include everyone. Like some people were friends before. There were modem parties where people from the boards got together, whether it was a software swap or a party.

Ronda: There weren't many, were there?

Michael: Well, what happened was the main person who had the parties was from a TAG board in Taylor. He had his computer stolen after the 2nd or 3rd party. So he stopped holding them. Then there were multi-user boards. There was MNET which was a multi-user. The general age of the users on MNet was older than on the other single-user BBS's. And it was more serious. It was more a UNIX board. It was a different bunch. It was not the home but the people in school, in Ann

Arbor. It seemed like the multi-user boards made it easier to hold parties because users could chat live one-on-one. And when AMUSERS (a multi-user board) closed down, I didn't get on other multi-users that were like AMUSERS. Some people already were friends but you didn't end up doing much so it was a little disappointing. Cause it didn't seem like there was any – it didn't get anywhere – it was just online so that was a little disconcerting. It was disappointing because that was where I had found more intellectual people but it didn't go anywhere. And things like CompuServe cost a lot of money. There's CompuServe, there's Delphi, there's Geni, there's PC Link, there's Q-link, there's a couple of services but they all cost money, so that's hard to deal with. And then there are bigger boards that exist. But they all cost money. There's the WELL. That's in California. You also pay per hour like CompuServe. So it's harder to be on. It's like MNet. It's the same software as MNet. And maybe I did find it disappointing. It used to be there would be lots of new BBS's popping up. But they were interesting. And now there still are lots of new BBS's popping up. But they're silly. So it's gone downhill a little bit. And also BBS's are similar to the CB or the Ham radio in that people voice their opinions, or have discussions or chat or there used to be DDial's – all they were were multi-user, people chatting, but they were 300 baud so they were super slow. Some of those you had to acquire membership. But they were linked up across the country. There were things called LINKS that would connect you to other DDials around the country. So that way you could talk to people.

Somehow the thing about BBS's was it was the ultimate vehicle of Free Speech, uncensored speech. For the most part things were not censored. What you posted was left alone. It was like everyone's Letter to the Editor was allowed to be printed. There would be letters debating other previous letters. Different Sys-Ops had different rules and some would delete messages that contained profanity or were only personal attacks or something. BBS's are the greatest form of free speech. The problem was you needed a modem and a computer to get into it. So it's not as free as it might be, but compared to the newspapers, the newspapers print what they choose, whereas on BBS's everything is printed, everything is published. It's more of a dynamic medium than a static medium because depending on the board there's different forms of

dealing with messages. For example, some boards after the first 50 messages go by, the first message is deleted, so it's a dynamic thing. Unless somebody prints out a copy or saves it to disk, it doesn't stay static. Like on MNet, things aren't deleted. They are deleted when the message sys-op of the area decides no one is interested anymore. That's more of a choice method of deletion, than where it deletes messages or the new one pops in, the old one pops out and it's deleted. And even depending on what happens, it's still an important medium.

There was, for example, just a debate about the war against IRAQ on BBS's. Usually you didn't see where there was dissent. Whereas on the computer, if people wanted to, they could debate it and there was debate about it. A free medium. It's open access. Not closed. It's also a field where the hobbyist still exists. There are people who develop ways of using the modem, whether it's different compression techniques where you can send more and larger files quicker, or whether it's different file protocols that send them faster over phone lines. Those are constantly developing. That is a hobbyist frontier now. Maybe there are less people than when the computer started out. But it still exists. It's a frontier that's not closed up yet. It's not definite yet. New things are continuing to come out. For example, higher speed chips for the serial ports in the computer so that the computer can talk to the modem at a higher speed and everything.

(To be continued in next issue)

One Line Program In BASIC

```
10 FOR I=38 TO 255:PRINT"CHR$(";I;") = ";CHR$(I);" ";:NEXT
```

(Editor's Note: We welcome other one line programs from readers.)

Computers For The People A History
Part III

And it was indeed. “In January, 1975, six months after the introduction of [Titus’ computer-ed] the Mark-8, *Popular Electronics* published the first installment of a two-part article on a much more sophisticated computer, the Altair 8800,” writes Stan Augarten (in *Bit by Bit*, NY, 1984). “The Altair was the first... full fledged personal computer on the market.” And it was available in kit form for $395,or in assembled form for $650. “Thousands of orders poured into MITS,” the manufacturer of the Altair, after the article was published. The article was featured on the front cover of the magazine: “Project Breakthrough: World’s First Minicomputer Kit to Rival Commercial Models Altair 8800. Save over $1000.” (p.270-273)

The firm was overwhelmed with orders. Augarten explains, “If you wanted to get the Altair to do something...you had to write a program in machine code and enter it, bit by bit, via the toggle switches on the front panel.” It only had a 256-byte memory, too small to do much with. “As a result,” writes Augarten, “all you could really do with the Altair was play with it, and one of its first programs was a game that generated increasingly complicated patterns of lights on the front panel to be duplicated by the players.” (p. 274)

What the Altair needed was an interpreter – a program that would allow the machine’s users to write programs in a simple computer language like BASIC. Paul Allen, was a programmer working in the Boston area. He had been a computer hacker during his teen years. He was strolling through Harvard Square one day when he noticed the January, 1975 issue of *Popular Electronics* on the newsstand. He bought the magazine and went to visit his friend and fellow hacker, William Gates, who was a freshman at Harvard. Allen is reported to have greeted Gates waving the article in front of Gates’ face and saying “Look, it’s going to happen! I told you this was going to happen! And we’re going to miss it.!” (Paul Freiberger and Michael Swaine, *Fire in the Valley*, Berkeley, 1984, p. 141) Freiberger and Swaine, describe what followed:

“Gates had to admit that his friend was right; it sure looked as

though the `something' they had been looking for had found them. He immediately phoned MITS [*producer of the Altair-ed*], and claimed that he and his partner had a BASIC language usable on the Altair. When Ed Roberts [*owner of MITS - ed*] who had heard a lot of such promises asked Gates when he could come to Albuquerque to demonstrate it, Gates looked at his childhood friend, took a deep breath, and said, 'Oh, in two or three weeks.' Gates put down the receiver, turned to Allen and said: 'I guess we should go buy a manual.' They went straight to an electronics shop and purchased Adam Osborne's manual on the 8080." (Ibid., p. 141)

Allen and Gates had committed themselves to producing a BASIC language for a machine they had not yet even seen. But they were both computer hackers from their junior high school days when they had worked for companies looking for bugs in commercial programs. Freiberger and Swine describe what happened next:

"For the next few weeks, Gates and Allen worked day and night on the BASIC. As they wrote the program, they tried to determine the minimal features of an acceptable BASIC.... There was no established industry standard for BASIC or for any other software. There was no industry. By deciding themselves what BASIC required, Gates and Allen set a pattern for future software development that lasted for about six years. Instead of researching the market, the programmers simply decided, at the outset, what to put in." (pp. 141 - 142)

Freiberger and Swaine continue their description of those significant days:

"Both men threw themselves completely into the project, staying up late every night programming.... They were programming half-asleep sometimes. Once when Gates nodded off, head on the keys, he woke up suddenly, glanced at the screen, and immediately began typing. Paul Allen decided Bill must have been programming in his sleep and kept right on when he awoke." (p. 142)

Six weeks later, Allen flew to Albuquerque with their BASIC interpreter. On the plane, he realized they hadn't written a program to load the language into the computer. He quickly made up a program on some scrap paper. Roberts met him at the airport and drove him to his workplace. He loaded the program into the Altair and it worked. Allen

became MITS's software director. Gates soon dropped out of Harvard and became a freelance software writer. Gates and Allen later set up the Microsoft Corporation in Bellevue, Washington, now one of the largest software companies in the world.

"The Altair inspired many hobbyists to design their own computers," writes Augarten, p. 276. One of these was Stephen Wozniak. The ferment from the introduction of the Altair had led to the formation of computer clubs all over the country, including one in California's Silicon Valley. The atmosphere generated by the clubs inspired hobbyists and hackers like Stephen Wozniak and Steve Jobs to create and then market computers like the Apple. Woz, as Steve Wozniak is known, explains how the Apple was created. He bought an 8-bit microprocessor, the 6502 from MOS technology (now part of Commodore), and he wrote a BASIC interpreter for it. The device wasn't as powerful as the Altair, but it was cheaper and less complicated. Then he built a circuit board and he and Steve Jobs set out to market it as the Apple Computer. Later in 1977 they introduced the Apple II with 16K of Ram for $1,195. Woz who loved to play games had designed the computer with that purpose in mind. The Apple II provided a commercial personal computer that people and schools could begin to afford. But crucial to the development of the Apple computer and to all the other advances made during this fruitful period in the development of the personal computer was the role played by the Homebrew Computer Club. Woz went to the first meeting, in March, 1975. About 30 people showed up. But fueled by the excitement generated by the Altair, the club expanded rapidly. Soon the Club had 500 members. Meetings divided into a "random-access" period during which the floor was thrown open to anyone who had anything to say, and a "mapping" period when the audience broke up into small groups devoted to common concerns.

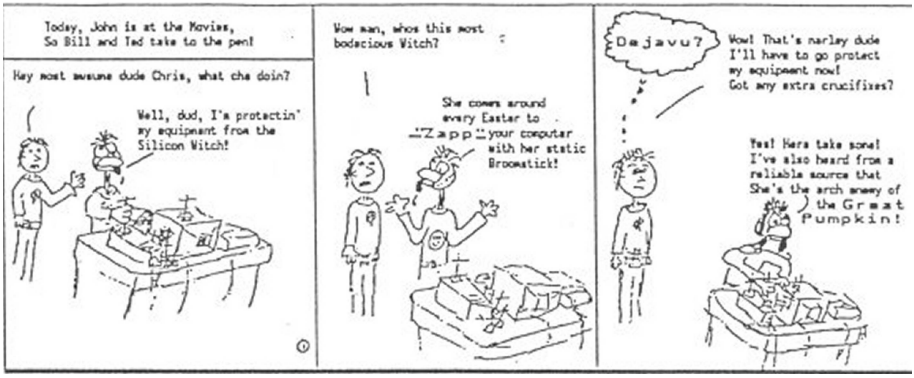
Lee Felsenstein is the person credited with making the club the important gathering it became. Felsenstein had been active in the Free Speech Movement at the University of California at Berkeley. He had been arrested in 1964 along with 755 other students for sitting in at the University. He had worked on the *Berkeley Barb* and the *Berkeley Tribe*, two newspapers of the student anti-war movement of the 1960's.

Lee Felsenstein, like Ted Nelson whose book *Computer Lib* became known as the Common Sense (a la Tom Paine) of the Technological Revolution, was one of a group of technological revolutionaries who were products of the radical 1960s. "A surprising number of them," write the authors of *Fire in the Valley,* "held political views that would have shocked the local Rotary Club and almost all had no love for IBM and the computer establishment." (Freiberger and Swaine, p. 108)

Keith Britton, another early member of the Homebrew Club, recalls the atmosphere prevalent during the early days of the personal computer movement. "There was a strong feeling," he writes, "that we were subversives. We were subverting the way the giant corporations had run things. We were upsetting the establishment forcing our mores into the industry. I was amazed that we could continue to meet without people arriving with bayonets to arrest the lot of us." (Ibid., p. 104)

Britton saw himself and the other members of the Homebrew Club as "pivotal in an equivalent of the industrial revolution but more profoundly important to the human race." (Ibid., p. 108)

(To be continued)



Pascal Program
Bridge_opening_bid_simulator
Uses Printer

```
(*******************************************************)
(*            Author      : Ian Carsten                *)
(*                                                     *)
(*            Last Modified  : 11-18-90                *)
(*                                                     *)
(* Input: menu choices 1-8 (integer), number or letter *)
(* of card (char) and first letter of suit (char)      *)
(*                                                     *)
(* Output: menu of options, 13-card hand, total points *)
(* of hand, and opening bid based on points and        *)
(* distribution of cards in hand                       *)
(*                                                     *)
(* Purpose: To simulate the opening bid in the card    *)
(* game of bridge as an aid in learning the strategy   *)
(* of assigning points and giving the opening bid      *)
(*                                                     *)
(* Features: In the change card option, if the user    *)
(* tries to remove a card which is not in the hand or  *)
(* tries to add a card which is already in the hand,   *)
(* then an appropriate error message is printed and the *)
(* particular add or delete routine is repeated-until  *)
(* a valid card change is input. Further, at start-up, *)
(* since the handarray is empty, if the user selects   *)
(* menu options 2,4,5,6, or 7 an empty hand message is *)
(* printed and the menu is presented to prevent mean-  *)
(* ingless execution and/or output. Although this      *)
(* program prevents copying an empty hand (52 ' 0 '    *)
(* --string3) to a file, nonetheless, an empty hand has *)
(* been stored under 'hand0.dat' to demonstrate what   *)
(* would happen if an empty hand were stored using DOS *)
(* commands and someone then tried to read the empty   *)
(* hand into the handarray. In bidding the case        *)
(* 13 <= points <= 18 and at least 5 cards in 1 or two *)
(* suits, the bid is: 1 of the 5-card suit with the    *)
(* greatest number of points. If each of 2 5-card suits *)
```

```
(* have the same number of points, then the bid is 1   *)
(* of the first 5-card suit in the order Spade, Heart,  *)
(* Diamond, Club. Since Givopeningbid calls             *)
(* Showpointsofhand, the bid can be used without first *)
(* getting the points.                                  *)
(******************************************************)
{Declaration of global variables} type
    string3 = string[3];
    stringarray = array[1..4, 1..13] of string3;
  var
    choice : integer; {user's choice from menu}
    emptyhand : boolean; {emptyhand = true
    prevents meaningless output}
    card : string3; {representation of card
    in handarray}
    handarray : stringarray; {4 x 13 array
    representation of card hand}
    points : integer; {points of hand, used
    to determine bid}
    cardsinsuit : array [1..4] of integer;
    {number of cards in suit}
    pointsinsuit : array [1..4] of integer;
    {number of points in suit}
(******************************************************)
(* Name : Initializearray                              *)
(*                                                     *)
(* Purpose : Initialize all elements of handarray to   *)
(* the placeholder ' 0 ' (string3)                     *)
(*                                                     *)
(* Preconditions : none                                *)
(*                                                     *)
(* Postconditions : all 52 elements of handarray are   *)
(* ' 0 ' (string3)                                     *)
(******************************************************)
    procedure Initializearray(var
    handarray{input/output} :stringarray);
      var
        row : integer;
        col : integer;
      begin {Initializearray}
        for row := 1 to 4 do {set each element of
        handarray to 0 (string3)}
          begin {row-loop}
            for col := 1 to 13 do
```

```
              begin {col-loop}
                handarray[row, col]
                := ' 0 ';
              end; {col-loop}
          end; {row-loop}
      end; {Initializearray}
(*******************************************************)
(* Name : Testforemptyhand                             *)
(*                                                     *)
(* Purpose : Determines if handarray is empty          *)
(*                                                     *)
(* Preconditions : handarray is either empty (all      *)
(* elements are ' 0 ' ) or it contains 13 cards and    *)
(* 39 placeholding ' 0 '                               *)
(*                                                     *)
(* Postconditions : emptyhand (boolean) is true if     *)
(* hand empty otherwise emptyhand is false             *)
(*******************************************************)
    procedure Testforemptyhand(handarray{input} :
    stringarray;
     var emptyhand{output} :boolean);
      var
        row : integer;
        col : integer;
      begin
        emptyhand := true; {initialize} for row := 1 to 4
        do
          begin {row-loop}
            for col := 1 to 13 do
              begin {col-loop}
                if handarray[row, col] <> ' 0 ' then
                emptyhand := false
              end; {col-loop}
          end; {row-loop}
      end; {Testforemptyhand}
(*******************************************************)
(* Name : Readcardandsuit                              *)
(*                                                     *)
(* Purpose : reads the two characters of user-input    *)
(* card and suit which user wishes to delete/add       *)
(*                                                     *)
(* Preconditions : user has input card (1 or, in the   *)
(* case of a '10', 2 characters ), a space, and the    *)
(* first letter of the suit, each of type char         *)
```

```
(*                                                    *)
(* Postconditions : charcard and charsuit have been    *)
(* assigned the upper case of user-input card and      *)
(* suit respectively                                  *)
(******************************************************)
    procedure Readcardandsuit(var
    charcard{input/output} : char;
      var charsuit{input/output} :char); var
        space : char; charzero : char;
      begin
        read(charcard); if charcard <> '1' then
          readln(space, charsuit);
        if charcard = '1' then {userinput card is a '10'}
          readln(charzero, space, charsuit);
        charcard := upcase(charcard);
        charsuit := upcase(charsuit);
      end;
(******************************************************)
(* Name : Displayhand                                 *)
(*                                                    *)
(* Purpose : To print the hand currently stored in    *)
(* handarray, with rows right-justified, lowest card to *)
(* highest from left to right (reverse order of array), *)
(* row 1 = Spades, row 2 = Hearts, row 3 = Diamonds,  *)
(* and row 4 = Clubs                                  *)
(*                                                    *)
(* Preconditions : handarray has a nonempty hand of 13  *)
(* unique cards and 39 placeholders (' 0 ') each      *)
(* of type string3                                    *)
(*                                                    *)
(* Postconditions : the hand has been printed         *)
(******************************************************)
    procedure Displayhand(handarray
    {input} : stringarray);
      type
        string10 = string[10];
      var
        row  : integer; col  : integer; suit : string10;
      begin {Displayhand}
      Testforemptyhand(handarray
      {input}, emptyhand{output});
      if (not emptyhand) then
        begin {execute Displayhand--
        nonempty hand}
```

```
        for row := 1 to 4 do
          begin {row-loop}
            case row of {assign suit
             based on its row number}
            1 : suit := 'Spades    ';
            2 : suit := 'Hearts    ';
            3 : suit := 'Diamonds  ';
            4 : suit := 'Clubs     ';
            end; {case}
          write(suit);
          write(Lst, suit);
          for col := 13 downto 1 do
            begin
              if handarray[row, col] <> ' 0 ' then
                begin {if}
                write(hand array [row, col]);
                   write(Lst, hand array[row, col]);
                end; {if}
            end; {col-loop} writeln; writeln(Lst);
          end; {row-loop}
        writeln; writeln(Lst);
      end {execute Displayhand--nonempty hand} else
        begin {else}
          writeln('Current hand is empty. Choose 1) or 3)
        from menu to '); writeln('get a nonempty hand.');
          writeln; writeln(Lst, 'Current hand is empty.
          Choose 1) or 3) from menu to ');
          writeln(Lst, 'get a nonempty hand.');
          writeln(Lst);
        end; {else}
      end; {Displayhand}
(*******************************************************)
(* Name : Convertcard                                  *)
(*                                                     *)
(* Purpose : Converts card and suit (char)input by user *)
(* into their string equivalents. Also, it assigns the *)
(* row and col (integer) of the card (string3) in hand array *)
(*                                                     *)
(* Preconditions : charcard and charsuit have been     *)
(* assigned values (char)                              *)
(*                                                     *)
(* Postconditions : row, col (integer), and card       *)
(* (string3) have been assigned values                 *)
(*******************************************************)
```

```
procedure Convertcard( charcard
{input} : char;
   charsuit{input} : char;
      var row{output} : integer;
      var col{output} : integer;
      var card{output} : string3);
  begin {Convertcard}
    if upcase(charsuit) = 'S'
    then {assign row number based
    on suit input }
      row := 1; {by user}
    if upcase(charsuit) = 'H' then
      row := 2;
    if upcase(charsuit) = 'D' then
      row := 3;
    if upcase(charsuit) = 'C' then
      row := 4;
    if charcard = 'A' then {assign
    col number and card based on
    charcard}
      begin {input by user}
        card := ' A ';
        col := 1;
      end;
    if charcard ='K' then
      begin
        card := ' K ';
        col := 2;
      end;
    if charcard = 'Q'then
      begin
        card := ' Q ';
        col := 3;
      end;
    if charcard = 'J' then
      begin
        card := ' J ';
        col := 4;
      end;
    if charcard = '1' then
      begin
        card := '10 ';
        col := 5;
      end;
```

```
      if charcard = '9' then
        begin
          card := ' 9 ';
          col := 6;
        end;
        if charcard = '8' then
        begin
          card := ' 8 ';
          col := 7;
        end;
      if charcard = '7' then
        begin
          card := ' 7 ';
          col := 8;
        end;
      if charcard = '6' then
        begin
          card := ' 6 ';
          col := 9;
        end;
      if charcard = '5' then
        begin
          card := ' 5 ';
          col := 10;
        end;
      if charcard = '4' then
        begin
          card := ' 4 ';
          col := 11;
        end;
      if charcard = '3' then
        begin
          card := ' 3 ';
          col := 12;
        end;
      if charcard = '2' then
        begin
          card := ' 2 ';
          col := 13;
        end;
    end; {Convertcard}
(*******************************************************)
(* Name : Translate                                    *)
(*                                                     *)
```

```
(* Purpose : Translates cardnum (integer) into its     *)
(* equivalent card (string3) and assigns row (integer) *)
(* and col (integer) which determines its appropriate  *)
(* location in handarray                               *)
(*                                                     *)
(* Preconditions : cardnum has been                    *)
(* assigned an integer value from 1-52                 *)
(*                                                     *)
(* Postconditions : row has been assigned an integer   *)
(* value from 1-4 and col has been assigned an integer *)
(* value from 1-13 and card (string3) has been         *)
(* assigned a value                                    *)
(*******************************************************)
  procedure Translate(cardnum :
  integer;
     var row : integer;
     var col : integer);
    begin {Translate}
      case cardnum of {assign row
      based on cardnum}
         1..13 : row := 1;
        14..26 : row := 2;
        27..39 : row := 3;
        40..52 : row := 4;
        end; {case-assign row}
      case cardnum of {assign col and
      card based on cardnum}
         1,14,27,40 : begin
                      col  :=  1;
                      card := ' A '
                      end;
         2,15,28,41 : begin
                      col  :=  2;
                      card := ' K '
                      end;
         3,16,29,42 : begin
                      col  :=  3;
                      card := ' Q '
                      end;
         4,17,30,43 : begin
                      col  :=  4;
                      card := ' J '
                      end;
         5,18,31,44 : begin
```

```
                    col  := 5;
                    card := '10 '
                    end;
         6,19,32,45 : begin
                    col  := 6;
                    card := ' 9 '
                    end;
         7,20,33,46 : begin
                    col  := 7;
                    card := ' 8 '
                    end;
         8,21,34,47 : begin
                    col  := 8;
                    card := ' 7 '
                    end;
         9,22,35,48 : begin
                    col  := 9;
                    card := ' 6 '
                    end;
        10,23,36,49 : begin
                    col  := 10;
                    card := ' 5 '
                    end;
        11,24,37,50 : begin
                    col  := 11;
                    card := ' 4 '
                    end;
        12,25,38,51 : begin
                    col  := 12;
                    card := ' 3 '
                    end;
        13,26,39,52 : begin
                    col  := 13;
                    card := ' 2 '
                    end;
        end; {case-assign col and
        card}
    end; {Translate}
(*****************************************)
(* Name : Showmenu                       *)
(*                                       *)
(* Purpose : To print a menu of options  *)
(* and prompt the user for his choice    *)
(*                                       *)
```

```
(* Preconditions : none                    *)
(*                                         *)
(* Postconditions : menu and prompt        *)
(* have been printed                       *)
(*******************************************)
  procedure Showmenu;
    begin {Showmenu}
      writeln('1) Generate a random
      hand');
      writeln('2) Save the current
      hand (to a text file)');
      writeln('3) Read a hand (from a
      text file)');
      writeln('4) Change a card in
      the current hand');
      writeln('5) Display the current
      hand');
      writeln('6) Show total points
      of the hand');
      writeln('7) Give the opening
      bid with the current hand');
      writeln('8) Quit');
      writeln;
      write('Enter your choice ');
      writeln(Lst,'1) Generate a
      random hand');
      writeln(Lst,'2) Save the cur-
      rent hand (to a text file)');
      writeln(Lst,'3) Read a hand
      (from a text file)');
      writeln(Lst,'4) Change a card
      in the current hand');
      writeln(Lst,'5) Display the
      current hand');
      writeln(Lst,'6) Show total
      points of the hand');
      writeln(Lst,'7) Give the open-
      ing bid with the current
      hand');
      writeln(Lst,'8) Quit');
      writeln(Lst);
      write(Lst,'Enter your choice
       ');
    end; {Showmenu}
```

```
(*********************************************************)
(* Name : Generatehand                                   *)
(*                                                       *)
(* Purpose : To generate 13 unique random integers from *)
(* 1 to 52, translate them into string3 representing a *)
(* hand of cards in the game of bridge and store the   *)
(* strings in in a 4 x 13 array representing the card   *)
(* hand according to the following scheme: top to      *)
(* bottom: row 1 = Spades, row 2 = Hearts, row 3 =     *)
(* Diamonds, row 4 = Clubs left to right: col 1 = ' A ',     *)
(* col 2 = ' K ',..., col 13 = ' 2 '. Also, all positions    *)
(* not holding card will have the string ' 0 ' (string3)*)
(* as a placeholder                                      *)
(*                                                       *)
(* Preconditions : handarray either holds 52 ' 0 '     *)
(* placeholders or it holds 13 cards and 39 ' 0 '      *)
(* placeholders, each of type string3                    *)
(*                                                       *)
(* Postconditions : handarray has 13 unique cards      *)
(*(string3) and 39 ' 0 ' placeholders, each of type string3   *)
(*********************************************************)
  procedure Generatehand(var hand-
  array : stringarray);
    type
      string10 = string[10];
    var
      colindex : integer;
      cardsinhand : integer; {number
      of cards in hand}
      inhand : boolean; {true if card
      being checked is already in
      hand}
      cardnum : integer; {random
      number of card}
      row : integer;
      col : integer;
      suit : string10;
    begin {Generatehand}
      Initializearray(handarray);
      {to ' 0 '}
      cardsinhand  := 0; {initialize
      number of cards in hand}
      while cardsinhand < 13 do
        begin {while}
```

```
          inhand := false; {reset to
          test next cardnum}
          cardnum := random(52) + 1;
          Translate(cardnum, row,
          col); {determines row, col}
          for colindex := 1 to 13 do
            begin {colindex-loop}
            if handarray[row, col-
            index] = card then
              inhand := true;
            end; {colindex-loop}
            if (not inhand) then
              begin {if-then}
                handarray[row,col] :=
                card; {add card to
                handarray}
                cardsinhand := cards
                inhand + 1;
              end; {if-then}
      end; {while}
    end; {Generatehand}
(*****************************************)
(* Name : Savehandtofile                 *)
(*                                       *)
(* Purpose : saves cardhand to filename  *)
(* which user specifies just prior to save*)
(*                                       *)
(* Preconditions : card hand exists and  *)
(* may be empty or nonempty              *)
(*                                       *)
(* Postconditions : nonempty card hand in *)
(* handarray has been saved to user-input *)
(* filename                              *)
(*****************************************)
  procedure Savehandtofile(handarray
  : stringarray);
   type
     string20 = string[20];
   var
     row : integer;
     col : integer;
     outfile : text;
     cardfile : string20;
    begin {Savehandtofile}
```

```
      Testforemptyhand(handarray
      {input}, emptyhand{output});
      if (not emptyhand) then
      begin {execute Savehandtofile-
      -nonempty hand}
        write('What is the name of
        the card file? ');
        write(Lst, 'What is the name
        of the card file? ');
        readln(cardfile);
        writeln(Lst, ' ', cardfile);
        assign(outfile, cardfile);
        rewrite(outfile);
        for row := 1 to 4 do
          begin
            for col := 1 to 13 do
              begin {col-loop}
                write(outfile, hand-
                array[row, col]);
              end; {col-loop}
            writeln(outfile);
          end; {row-loop}
        close(outfile);
      end {execute Savehandtofile--
      nonempty hand}
      else
        begin
        writeln('Current hand is
        empty. Choose 1) or 3) from
        menu to ');
        writeln('get a nonempty
        hand.');
        writeln;
        writeln(Lst, 'Current hand is
        empty. Choose 1) or 3) from
        menu to ');
        writeln(Lst, 'get a nonempty
        hand.');
        writeln(Lst);
        end;
    end; {Savehandtofile}
(****************************************)
(* Name : Readhandfromfile              *)
(*                                      *)
(* Purpose : reads hand from user-input *)
(* filename to handarray                *)
(*                                      *)
(* Preconditions : user-input filename  *)
```

```
(* exists and holds a card hand           *)
(*                                         *)
(* Postconditions : cardhand in filename  *)
(* has been copied to handarray           *)
(*****************************************)
    procedure Readhandfromfile(var
    handarray : stringarray);
      type
        string20 = string[20];
      var
        row : integer;
        col : integer;
        outfile : text;
        cardfile : string20;
      begin {Readhandfromfile}
        write('What is the name of
        the file? ');
        write(Lst, 'What is the name
        of the file? ');
        readln(cardfile);
        write(Lst, ' ', cardfile);
        assign(outfile, cardfile);
        reset(outfile);
        for row := 1 to 4 do
          begin {row-loop}
            for col := 1 to 13 do
              begin {col-loop}
                read(outfile, hand-
                array[row, col]);
              end; {col-loop}
            readln(outfile);
          end; {row-loop}
        writeln(Lst);
        close(outfile);
        Testforemptyhand(handarray
        {input}, emptyhand{output});
        if (emptyhand) then
          begin {if-emptyhand}
            writeln('File ',cardfile,
            ' is empty. Choose 1) or
            3) (with');
            writeln('a nonempty file)
            from menu to get a non-
            empty hand.');
            writeln;
            writeln(Lst, 'File ',
            cardfile, ' is empty.
            Choose 1) or 3) (with');
            writeln(Lst, 'a nonempty
```

```
            file) from menu to get a
            nonempty hand.');
            writeln(Lst);
          end; {if-emptyhand}
      end; {Readhandfromfile}
(*******************************************************)
(* Name : Changecard                                   *)
(*                                                     *)
(* Purpose : first deletes a  userspecified card from  *)
(* handarray, then adds one to the array               *)
(*                                                     *)
(* Preconditions : handarray exists and may be empty or *)
(* nonempty (but Changecard will not execute           *)
(* if handarray empty)                                 *)
(*                                                     *)
(* Postconditions : handarray has a unique nonempty    *)
(* hand (if Changecard executed)                       *)
(*******************************************************)
  procedure Changecard(var handarray
  : stringarray);
    type
      string8 = string[8];
    var
      charcard    : char;
      space       : char;
      charzero    : char;
      charsuit    : char;
      row         : integer;
      col         : integer;
      inhand      : boolean;
      suit        : string8;
    begin {Changecard}
      Testforemptyhand(handarray
      {input}, emptyhand{output});
      if (not emptyhand) then
        begin {execute Changecard--
        nonempty hand}
{delete chosen card provided it is in hand}
          inhand := false;
          while (not inhand) do
           begin {while-not inhand}
            inhand := false;
            write('What card do you
            want to remove(card-
            <space>suit) ? ');
            write(Lst, 'What card do
            you want to remove(card-
            <space>suit) ? ');
            Readcardandsuit(charcard,
```

```
            charsuit);
            Convertcard(charcard,
            charsuit, row, col,
            card);
            case row of
              1 : suit := 'Spades';
              2 : suit := 'Hearts';
              3 : suit := 'Diamonds';
              4 : suit := 'Clubs';
              end; {case}
            writeln(Lst, card, ' ',
            charsuit);
            if (handarray[row, col] =
            card) then
            inhand := true;
            if (inhand) then
              begin {if-inhand}
                handarray[row, col]
                := ' 0 '; {delete
                card from hand}
             end {if-inhand}
        else
        begin {else}
          writeln(card, 'of ',  suit,
          ' is not in hand');
          writeln(Lst, card, 'of ',
          suit, ' is not in hand');
        end; {else}
    end; {while-not inhand}
{add chosen card if not already in hand}
    inhand := true;
    while inhand do
      begin {while-inhand}
        inhand := true;
        write('What card do you want
        to add(card<space>suit) ? ');
        write(Lst, 'What card do you
        want to add(card<space>suit)
        ? ');
        Readcardandsuit(charcard,
        charsuit);
        Convertcard(charcard, char-
        suit, row, col, card);
        case row of
          1 : suit := 'Spades';
          2 : suit := 'Hearts';
          3 : suit := 'Diamonds';
          4 : suit := 'Clubs';
        end; {case}
```

```
        writeln(Lst, card, ' ', char-
        suit);
        if (handarray[row, col] <>
        card) then
          inhand := false;
        if (not inhand) then
          begin {if}
            handarray[row, col] :=
            card; {add card to hand}
          end {if}
        else
        begin {else}
          writeln(card, 'of ',  suit,
          ' is already in hand');
          writeln(Lst, card, 'of ',
          suit, ' is already in
          hand');
        end; {else}
      end; {while-inhand}
      end {execute Changecard--non-
      empty hand}
      else
      begin {else}
        writeln('Current hand is
        empty. Choose 1) or 3) from
        menu to ');
        writeln('get a nonempty
        hand.');
        writeln;
        writeln(Lst, 'Current hand is
        empty. Choose 1) or 3) from
        menu to ');
        writeln(Lst, 'get a nonempty
        hand.');
        writeln(Lst);
      end; {else}
    end; {Changecard}
(*******************************************************)
(* Name : Showpointsofhand                             *)
(*                                                     *)
(* Purpose : calculates and prints the points of the hand   *)
(*                                                     *)
(* Preconditions : handarray has a unique nonempty hand of  *)
(* card (this procedure will not execute with an empty hand)*)
(*                                                     *)
(* Postconditions : points (integer) have been assigned *)
(*******************************************************)
procedure Showpointsofhand(handarray
{input} : stringarray; var points-
```

```
{output} : integer );
  var
    row : integer;
    col : integer;
    prevpoints : integer;
    begin {Showpointsofhand}
      Testforemptyhand(handarray-
      {input}, emptyhand{output});
      if (not emptyhand) then
        begin {execute Showpoints-
        ofhand--nonempty hand}
          points := 0;
          prevpoints := 0;
          for row := 1 to 4 do
            begin {row-loop}
              cardsinsuit[row] := 0;
              for col := 1 to 13 do
                begin {col-loop}
                  if handarray[row,
                  col] = ' A ' then
                    points := points
                    + 4;
                  if handarray[row,
                  col] = ' K ' then
                    points := points
                    + 3;
                  if handarray[row,
                  col] = ' Q ' then
                    points := points
                    + 2;
                  if handarray[row,
                  col] = ' J ' then
                    points := points
                    +1;
                  if handarray[row,
                  col] <> ' 0 ' then
                    cardsinsuit[row]
                    := cardsinsuit-
                    [row] + 1;
                end; {col-loop}
              case cardsinsuit[row]
              of
                0 : points := points
                + 2;
                1 : points := points
                + 1;
                end; {case}
              pointsinsuit[row] :=
              points - prevpoints;
```

```
            prevpoints := points;
          end; {row-loop}
        if choice <> 7 then {choice
        = 7, points passed to
        Giveopeningbid}
          begin {if-choice <> 7}
            if points = 1 then
              begin {if-points = 1}
                writeln(points, '
                point');
                writeln(Lst,
                points, ' point');
              end {if-points = 1}
            else
            begin {else}
              writeln(points, '
              points');
              writeln(Lst, points,
              ' points');
            end; {else}
          end; {if--choice <> 7}
      end {execute Showpointsof-
      hand--nonempty hand}
    else
      begin {else}
        writeln('Current hand is
        empty. Choose 1) or 3) from
         menu to ');
        writeln('get a nonempty
        hand.');
        writeln;
        writeln(Lst, 'Current hand
        is empty. Choose 1) or 3)
        from menu to ');
        writeln(Lst, 'get a
        nonempty hand.');
        writeln(Lst);
      end; {else}
  end; {Showpointsofhand}
(*******************************************************)
(* Name : Giveopeningbid                               *)
(*                                                     *)
(* Purpose : computes and prints the opening bid       *)
(*                                                     *)
(* Preconditions : handarray has a unique nonempty hand *)
(* (will not execute with an empty hand)               *)
(*                                                     *)
(* Postconditions : opening bid has been computed and  *)
(* printed                                             *)
```

```
(*******************************************************)
    procedure Giveopeningbid(var
    points : integer);
      type
        string10 = string[10];
      var
        bid : string10;
        suit : string10;
        row : integer;
        col : integer;
        rowindex : integer;
        cardsinrow : integer;
        mostcardsinsuit : integer;
        max : integer;
      begin {Giveopeningbid}
        Testforemptyhand(handarray-
        {input}, emptyhand{output});
        if (not emptyhand) then
          begin {execute Giveopening-
          bid--nonempty hand}
            Showpointsofhand(hand-
            array{input}, points-
            {output});
            mostcardsinsuit := 0;
            cardsinrow := 0;
            for row := 1 to 4 do
              begin {row-loop}
                for col := 1 to 13 do
                  begin {col-loop}
                    if handarray[row,
                    col] <> ' 0 '
                    then
                      cardsinrow :=
                      cardsinrow + 1
                  end; {col-loop}
              if cardsinrow > most-
              cardsinsuit then
                begin
                  mostcardsinsuit :=
                  cardsinrow;
                  rowindex := row
                end; {if}
                cardsinrow := 0
              end; {row-loop}
{case: causes bid to be 1 of which- }
{ever 5-card suit has highest number}
{of points. If 2 5-card suits have  }
{same number of points, then suit of}
{lowest number rowindex will be bid }
```

```
case rowindex of
  1 : suit := 'Spade';
  2 : suit := 'Heart';
  3 : suit := 'Diamond';
  4 : suit := 'Club';
  end; {case}
if points < 13 then
  begin {bid-case 1}
    writeln('pass');
    writeln(Lst, 'pass');
  end; {bid-case 1}
if (13 <= points) and
(points <= 18) and (most-
cardsinsuit >= 5) then
  begin {bid-case 2}
    max := 0;
    for row := 1 to 4 do
      begin  {row-loop}
        if (cardsinsuit[row] >= 5)
        then
          begin {if}
            if pointsinsuit[row] >
             max then
              begin {if}
                max := pointsin-
                suit[row];
                rowindex := row;
              end; {if}
        case rowindex of
          1 : suit := ' Spade ';
          2 : suit := ' Heart ';
          3 : suit := ' Diamond ';
          4 : suit := ' Club ';
          end; {case}
        end; {if}
      end; {row-loop}
  writeln('1 ', suit);
  writeln(Lst, '1 ', suit);
  end; {bid-case 2}
if (13 <= points) and
(points <= 15) and (most-
cardsinsuit < 5) then
  begin {bid-case 3}
    writeln('1 club');
    writeln(Lst, '1 club');
  end; {bid-case 3}
if (16 <= points) and (points
<= 18) and (most cardsinsuit < 5)
then begin
```

```
          {bid-case4}
              writeln('1 no trump');
              writeln(Lst, '1 no trump');
            end; {bid-case 4}
          if 18 < points then
            begin {bid-case 5}
              writeln('2 clubs');
              writeln(Lst, '2 clubs ');
            end; {bid-case 5}
      end {execute Giveopeningbid--
      nonempty hand}
      else
        begin {else}
          writeln('Current hand is empty.
          Choose 1) or 3) from menu to ');
          writeln('get a nonempty hand.');
          writeln;
          writeln(Lst, 'Current hand is
          empty. Choose 1) or 3) from
          menu to ');
          writeln(Lst, 'get a nonempty
          hand.');
          writeln(Lst);
        end; {else}
      end; {Giveopeningbid}
(*****************************************)
  begin {main}
    randomize;
    choice := 0; {initialize to insure entry
    into while loop}
    Initializearray(handarray); while choice
    <> 8 do
      begin {while}
        Showmenu;
        readln(choice);
        writeln(Lst,choice);
        case choice of
          1 : Generatehand(handarray);
          2 : Savehandtofile(handarray);
          3 : Readhandfromfile(handarray);
          4 : Changecard(handarray);
          5 : Displayhand(handarray);
          6 : Showpointsofhand(handarray{in-
          put}, points{output});
          7 : Giveopeningbid(points);
        else {quit if choice is 8}
        end; {case}
      end; {while}
    end. {main}
```

Special Thanks to Tim Henderson for scanning the graphics and helping to produce this newsletter. (the Editors)

The opinions expressed in articles are those of their authors and not necessarily the opinions of the *Amateur Computerist* newsletter. We welcome submissions from a spectrum of viewpoints.

ELECTRONIC EDITION

ACN Webpage:

http://www.ais.org/~jrh/acn/

All issues of the *Amateur Computerist* are on-line.
Back issues of the *Amateur Computerist* are available at:
http://www.ais.org/~jrh/acn/Back_Issues/

All issues can be accessed from the Index at:
http://www.ais.org/~jrh/acn/NewIndex.pdf

The *Amateur Computerist* invites submissions. Articles can be submitted via e-mail: jrh@ais.org