DOCUMENT RESUME

ED 056 531 EM 009 392

TITLE	Project Solo; Newsletter Number Eleven.
INSTITUTION	Pittsburgh Univ., Pa. Dept. of Computer Science.
SPONS AGENCY	National Science Foundation, Washington, D.C.
PUB DATE	15 Feb 71
NOTE	16p.; See also ED 053 566
EDRS PRICE	MF-$0.65 HC-$3.29
DESCRIPTORS	*Computer Programs; *Computer Science Education; Grade 9; Information Retrieval; Programing; *Student Developed Materials
IDENTIFIERS	*Project Solo

ABSTRACT

An experimental 9th grade computer science syllabus is proposed. The syllabus would include the technical information needed for controlling and programing the computer in a number of modes and would preview some of the areas covered in the high school curriculum. A sample module of a topic not normally taught in high school--distance and error-correcting codes--is presented. Two student-authored programs also illustrate the ways in which the computer can explore areas outside the normal curriculum--a plotter routine for making artistic patterns and a program to use the Monte Carlo method to calculate the area under a curve. (JY)

PROJECT SOLO

AN EXPERIMENT IN REGIONAL COMPUTING *
FOR SECONDARY SCHOOL SYSTEMS

University of Pittsburgh • Department of Computer Science • Pittsburgh, Pennsylvania 15213

Newsletter No. 11

February 15, 1971

Computer Science Modules

Our recent experience indicates that there is a lot of potential for getting youngsters in the 9th, 8th, (or sometimes 7th) grade excited about exploring the world of learning with computers. The potential is magnified several times when the student controls the full power of the computer; when it is used by him, not on him. This means that we must provide the opportunity for these students to master control of the machine at an early age. The implications for increased sophistication of use in the upper grades is obvious.

To this end we are proposing that an experimental 9th grade "Computer Science" syllabus be prepared for testing next fall. Such a syllabus would include the technical information needed for controlling and programming the computer in a number of modes, with applications that "previewed" some of the things the student could learn more about in his high school years ahead. For many students this could mean a new sense of purpose in working at their education.

The computer science syllabus would also include topics not normally taught in high school which have a "fun" characteristic to them. A sample module illustrating such a topic (DISTANCE AND ERROR-CORRECTING CODES) is enclosed.

Two Student-Authored Programs

The two programs enclosed also illustrate topics outside the usual curriculum. The first is a plotter routine for making artistic patterns written by Chris Van Sickle, an 8th grade student who "sneaked" into our program. The second (Monte Carlo Integration) is by Mike Kaufmann who is a senior at Allderdice.

Information Retrieval

A preliminary list of module areas that come under the heading of information retrieval is included with this newsletter. Some of these modules could supplement mathematics courses. They all can be classified under computer science. Students who master these techniques can apply them to any subject.

*Supported in part by NSF grant GJ 1077

PRELIMINARY LIST OF MODULE AREAS UTILIZING INFORMATION RETRIEVAL

1. Set Theory: Basic Concepts
   a. Definition
   b. Examples
   c. Modification of Set (updating)

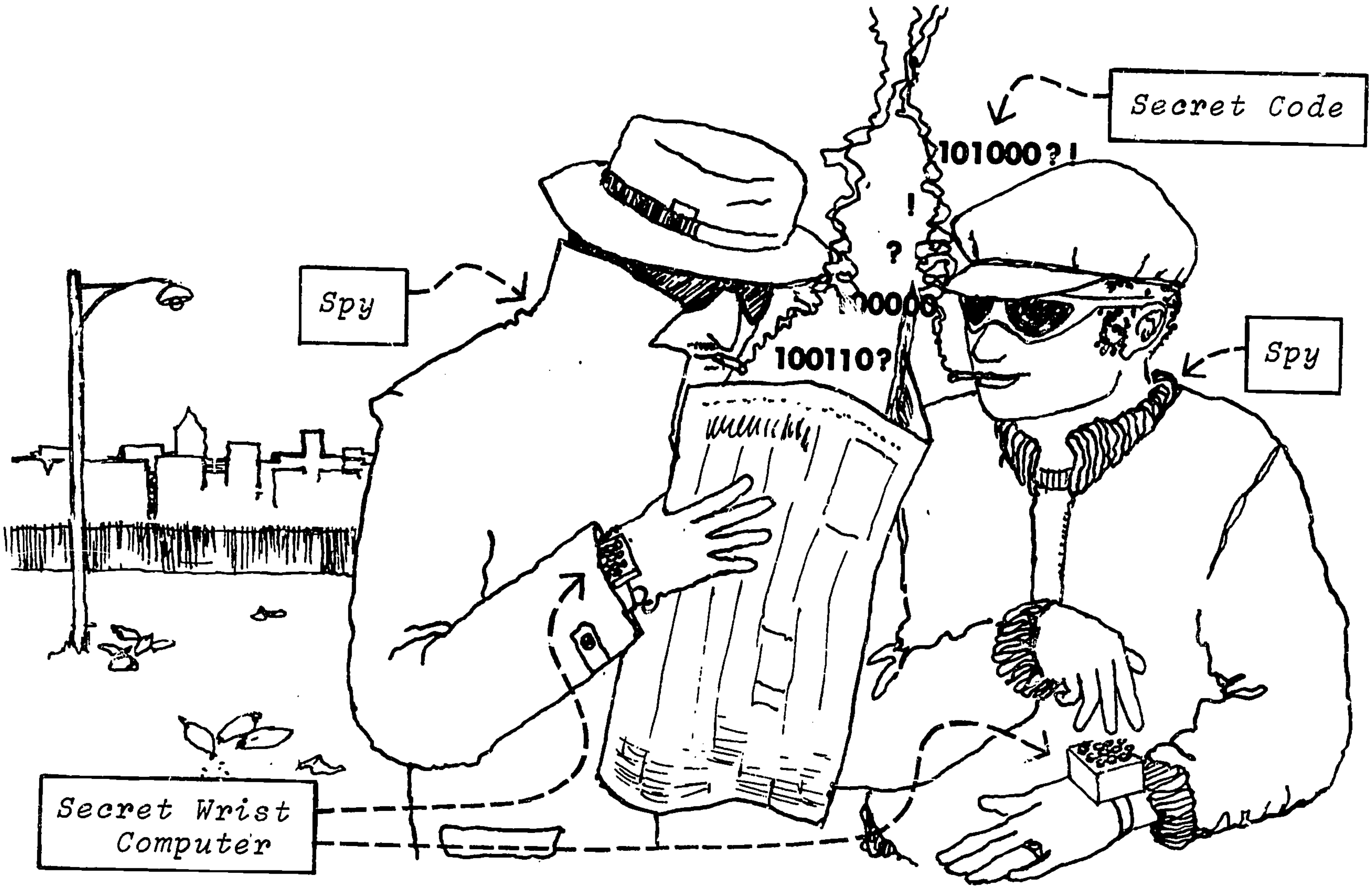
2. Operations on Sets
   a. Union
      Intersection
      Complement
      Commutative, associative and distributive laws
   b. Logical interpretation of operations

3. Information Retrieval Techniques
   a. Description
   b. Examples
   c. Building and updating file
      (1) selection of material
      (2) representation
      (3) structuring
      (4) set interpretation
   d. Searching
      (1) constructing query
      (2) query analysis
      (3) logical operations
      (4) iterating the search

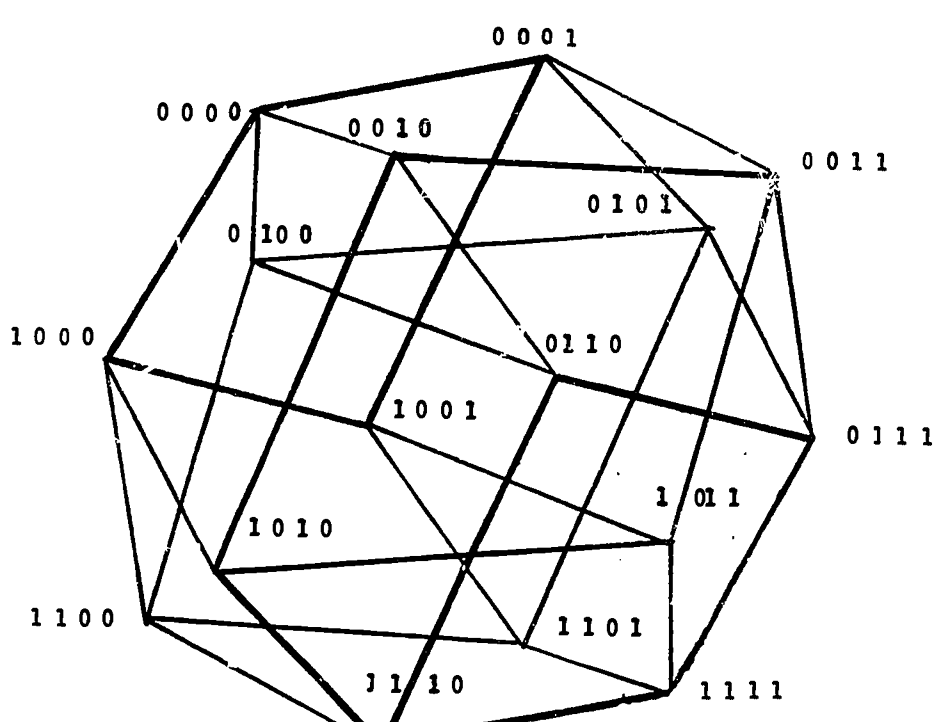
4. Information Retrieval: Advanced Concepts
   a. Relevance
   b. False drop analysis
   c. Modifying representation
   d. Syntactic and semantic problems
   e. Fact, reference, and information, retrieval

5. Guided Application of Information Retrieval
   a. Select subject matter (e.g. Apollo Moon Program)
   b. Student builds I.R. system using techniques given in 3 above.
   c. Student uses retrieval system to study some pertinent problem in the subject under consideration

6. New I.R. systems
   Student is to write and demonstrate his own fact retrieval system. (Some subjects might be professional football scores, prices of a given stock over the past 30 years, data on the planets, super-market prices, areas suggested by the school librarian, etc.)



Distance and Error-Correcting Codes



A binary code of length N is a string of N 0's or 1's. For example, if N = 3, all the possible binary codes are 000, 001, 010, 011, 100, 101, 110, and 111. We speak of these as 3 BIT codes (1 and 0 are called BITS).

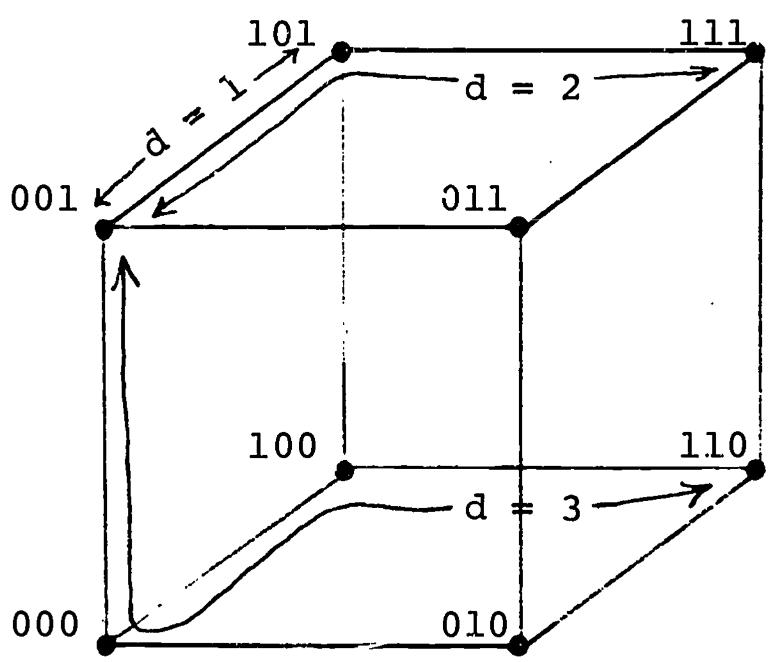
These codes could be used to represent eight objects of any sort--the members of a musical octet, the digits 0,1,2,3,4,5,6,7 in a computer, or the letters A,B,C,D,E,F,G,H.

Now for some intrigue

Let's assume that we wish to assign binary code names to the agents of STICK (Society to Increase Contact for Keeps), an international ring of glue thieves. Suppose we only have two agents but eight codes. Question: Can we assign codes so that:

(a) The computer will check code authenticity without knowing the correct codes.
(b) The computer can give the correct code even though the agent has deliberately changed one BIT (to throw off eavesdroppers).

To see how codes can be assigned to make this possible, let's place the codes at the vertices of a cube.



To be more precise, we should call the above figure a "3-dimensional cube". A picture of a "4-dimensional cube" (which has $2^4 = 16$ vertices) is shown on the cover. Thus we can associate a unique four-bit code with each vertex of a 4-D cube. Can you generalize this statement?

We will define the number of BITS by which two codes differ as the "DISTANCE"(=d) between these codes. Thus, for example, the distance between 001 and 101 is d = 1, the distance between 001 and 111 is d = 2, and the distance between 001 and 110 is d = 3. Math students: Is this a legal use of the word distance? Notice that our picture has been drawn so that "distance" between codes corresponds to the number of edges of the cube you would have to walk along to get from one vertex to the other.

Let's assign our two authentic agents the codes 001 and 110 (which are a distance of three from each other). Now suppose one agent walks up to another and says my code is 101.

(a) How can we tell if it is an authentic code? One way would be to simply compare it to the list of authentic codes! However, if there were to be very many codes, such a search of the authentic list would be time consuming. Besides, we don't want this authentic list stored in too many places! There is another way to check authenticity.

In our example the two authentic codes have the property that if we add the first and third BITS of the code we get 1, and this is <u>also</u> true if we add the second and third BITS.

```
0 0 1                1 1 0
  sum=1                sum=1
sum=1                sum=1
```

For all other codes this is false.

NOTE:	0 + 1 = 1 and 1 + 0 = 1 in binary arithmetic.
ALSO NOTE:	0 + 0 = 0, but 1 + 1 = 0 (with carry of 1)
FURTHER:	0 * 1 = 0 1 * 0 = 0 0 * 0 = 0 and 1 * 1 = 1

Thus the code 1 0 1 our agent gave is not authentic.

Since: 1 + 1 = 0

(b) Suppose our agent deliberately changes one BIT in his code when giving it verbally. Applying the above rule will <u>detect</u> the error, but can we figure out what the correct code should have been?

We can see the answer from our diagram. An authentic code which has only one BIT changed is distance d = 1 from the original correct code, but distance d = 2 from the <u>other</u> correct code. Thus 101 has to be corrected back to 001, <u>not</u> to 110.

Try to develop an algorithm for making corrections in codes which have 1 BIT in error: Here is how you might reason:

Let's call the 3 BITS: B1, B2, and B3.

FOR 101	B1 + B3 = 0	WRONG
	B2 + B3 = 1	RIGHT

$\therefore$ Change B1, Correct code is: 001

FOR 111	B1 + B3 = 0	WRONG
	B2 + B3 = 0	WRONG

$\therefore$ Change B3, Correct code is: 110 etc.

Problems: Write programs for your wrist computer to handle the following:

1. The Two-Agent Problem.
   Authentic Codes: 101 and 010

   INPUT: Any 3 BIT code which is either an authentic code, or which contains an error in 1 BIT.

   OUTPUT: The Message: "AUTHENTIC CODE"
   OR: "CODE IN ERROR
   CODE SHOULD BE_ _ _"

2. Can use of a 4 BIT code (see picture on cover) permit additional outputs for the above "Two-Agent" analysis program?

3. Four-Agent Problem.
   Authentic Codes are:

SMITH	0 0 0 0 0
BOND	1 1 1 0 0
SPIRO	0 0 1 1 1
JONES	1 1 0 1 1

   INPUT: Any code

   OUTPUT: The Message: "AUTHENTIC CODE"
   OR: "1 BIT ERROR--CORRECT CODE IS_ _ _"
   OR: "ERROR >=2 BITS--DOUBLE AGENT"

4. Here is a set of six BIT codes to play with: 000000, 000111, 111000, 110110, 011011, 101101. (NOTE: d>=3 for any two of these codes.)

Sample Solution--Problem 1

```
>LISTNH

10 S(1) = 0
20 S(2) = 1
35 PR."TWØ-AGENT PRØBLEM"
90 PR. "TØ END PRØGRAM, TYPE THE CHARACTERS END WHEN ASKED FØR A CØDE
"
95 PR." "
100 PR. "ENTER A 3-BIT CØDE."
105 INPUT B$
106 IF B$ = "END" GØTØ 185
110 GØSUB 200
115 S(3) = S1
120 S(4) = S2
125 IF (S(1) = S(3) AND S(2)=S(4)) GØTØ 180
126 IF (S(1) # S(3) AND S(2)#S(4)) GØTØ 140
129 X = 1-S(3)
130 IF (S(1) = X) GØTØ 150
134 X= 1-S(4)
135 IF (S(2) = X) GØTØ 160
140 B3$ = RIGHT(  STR(1-VAL(B3$)) ,1)
145 GØTØ 165
150 B1$ = RIGHT(  STR(1-VAL(B1$)), 1)
155 GØTØ 165
160 B2$ = RIGHT(  STR(1 - VAL(B2$)), 1)
165 PRINT" CØDE IN ERRØR"
170 PRINT" CØDE SHØULD BE ":B1$+B2$+B3$
172 GØTØ 95
175 STØP
180 PRINT" AUTHENTIC CØDE"
182 GØTØ 95
185 STØP
190 END
200 B1$ = LEFT(B$,1)
205 X$ = RIGHT(B$,2)
210 B2$ = LEFT(X$,1)
215 B3$ = RIGHT(B$,1)
220 S1 = VAL(B1$) + VAL(B3$)
225 S2 = VAL(B2$) + VAL(B3$)
230 IF S1 = 2 LET S1 = 0
235 IF S2 = 2 LET S2 = 0
240 RETURN
```

Problem 1 (cont.)

```
>RUN
TWØ-AGENT PRØBLEM
TØ END PRØGRAM, TYPE THE CHARACTERS END WHEN ASKED FØR A CØDE.

ENTER A 3-BIT CØDE.
?000
 CØDE IN ERRØR
 CØDE SHØULD BE 010

ENTER A 3-BIT CØDE.
?001
 CØDE IN ERRØR
 CØDE SHØULD BE 101

ENTER A 3-BIT CØDE.
?010
 AUTHENTIC CØDE

ENTER A 3-BIT CØDE.
?011
 CØDE IN ERRØR
 CØDE SHØULD BE 010

ENTER A 3-BIT CØDE.
?100
 CØDE IN ERRØR
 CØDE SHØULD BE 101

ENTER A 3-BIT CØDE.
?101
 AUTHENTIC CØDE

ENTER A 3-BIT CØDE.
?110
 CØDE IN ERRØR
 CØDE SHØULD BE 010

ENTER A 3-BIT CØDE.
?111
 CØDE IN ERRØR
 CØDE SHØULD BE 101

ENTER A 3-BIT CØDE.
?END
```

Sample Solution--Problem 3

```
>LISTNH

5 PR. "FØUR-AGENT PRØBLEM"
10 PR."TØ END PRØGRAM, TYPE THE CHARACTERS END WHEN ASKED FØR A CØDE"
12 PR. " "
13 VAR = ZERØ
15 PR."ENTER A 5-BIT BINARY CØDE"
20 INPUT B$
25 IF B$ = "END" GØTØ 250
30 B1$ = LEFT(B$,1)
35 X$ = RIGHT(B$,4)
40 B2$ = LEFT(X$,1)
45 X$ = RIGHT(B$,3)
50 B3$ = LEFT(X$,1)
55 X$ = RIGHT(B$,2)
60 B4$ = LEFT(X$,1)
65 B5$ = RIGHT(X$,1)
70 S(1) = VAL(B1$) + VAL(B2$)
75 S(2) = VAL(B4$) + VAL(B5$)
80 S(3) = VAL(B1$) + VAL(B4$)
85 S(4) = VAL(B2$) + VAL(B5$)
90 FØR I = 1 TØ 4
95 IF S(I) = 2 LET S(I) = 0
100 NEXT I
105 IF S(1) # 0 LET F1 = 1
110 IF S(2) = 0 GØTØ 125
115 IF F1 = 1 GØTØ 240
120 F2 = 1
125 IF VAL(B3$) # S(3) LET F3 = 1
130 IF VAL(B3$) # S(4) LET F4 = 1
135 K = F1 + F2 + F3 + F4
140 IF K = 0 GØTØ 230
145 IF K # 2 GØTØ 240
150 IF (F1 + F3) # 2 GØTØ 165
155 B1$ = RIGHT( STR(1 -VAL(B1$)),1)
160 GØTØ 215
165 IF (F1 + F4) # 2 GØTØ 180
170 B2$ = RIGHT( STR(1 - VAL(B2$)),1)
175 GØTØ 215
180 IF (F3 + F4) # 2 GØTØ 195
185 B3$ = RIGHT( STR(1 - VAL(B3$)),1)
190 GØTØ 215
195 IF (F2 + F3) #2 GØTØ 210
200 B4$ = RIGHT( STR(1 - VAL(B4$)),1)
205 GØTØ 215
210 B5$ =RIGHT( STR(1 - VAL(B5$)),1)
215 PR." "
220 PR." 1 BIT ERRØR---CØRRECT CØDE IS ":B1$+B2$+B3$+B4$+B5$
225 GØTØ 12
230 PR." AUTHENTIC CØDE"
235 GØTØ 12
240 PR." ERRØR = 2 BITS --- DØUBLE AGENT!"
245 GØTØ 12
250 STØP
260 END
```

```
>RUN
FØUR-AGENT PRØBLEM
TØ END PRØGRAM, TYPE THE CHARACTERS END WHEN ASKED FØR A CØDE

ENTER A 5-BIT BINARY CØDE
?00000
 AUTHENTIC CØDE

ENTER A 5-BIT BINARY CØDE
?10000

 1 BIT ERRØR---CØRRECT CØDE IS 00000

ENTER A 5-BIT BINARY CØDE
?00100

 1 BIT ERRØR---CØRRECT CØDE IS 00000

ENTER A 5-BIT BINARY CØDE
?00010

 1 BIT ERRØR---CØRRECT CØDE IS 00000

ENTER A 5-BIT BINARY CØDE
?11011
 AUTHENTIC CØDE

ENTER A 5-BIT BINARY CØDE
?11010

 1 BIT ERRØR---CØRRECT CØDE IS 11011

ENTER A 5-BIT BINARY CØDE
?10011

 1 BIT ERRØR---CØRRECT CØDE IS 11011

ENTER A 5-BIT BINARY CØDE
?10101
 ERRØR = 2 BITS --- DØUBLE AGENT!

ENTER A 5-BIT BINARY CØDE
?10110
 ERRØR = 2 BITS --- DØUBLE AGENT!

ENTER A 5-BIT BINARY CØDE
?END
```

THE MONTE CARLO METHOD

--A FUN STATISTICAL APPROACH TO CALCULATING THE AREA OF GEOMETRICAL FIGURES OR THE AREA UNDER A CURVE

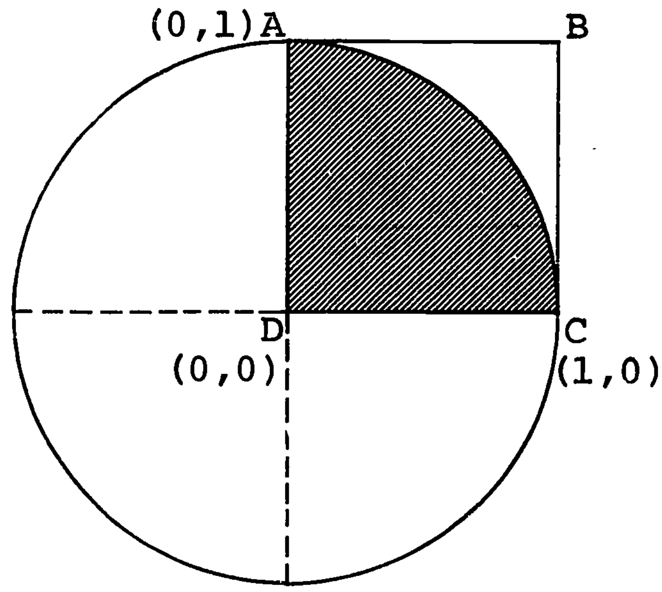
(by Michael Kaufman, Grade 12)

Method:

Set off a rectangle which includes the area to be calculated. Using the random number generator, try out many points included in the rectangle, to see if they are also included in the figure. The number of points included in the figure, divided by the total number of points tried, times the area of the rectangle, will approximate the unknown area.

Problem:

Find the area of a quarter of a unit circle using the Monte Carlo method.



Set off rectangle (square) ABCD so that it includes the quarter circle whose area is to be calculated. We will 'shoot' 500 points at the square. The points each have coordinates (x,y) where both x and y are between 0 and 1 (this puts the points within the square ABCD). The (x,y) coordinates will be found using the random number generator.

The equation of a unit circle is $x^2 + y^2 = r^2$, $r = 1$. Therefore $x^2 + y^2 = 1$, $y^2 = 1 - x^2$, $y = \sqrt{1 - x^2}$. (The $\sqrt{1 - x^2}$ eliminates all negative values of y so that this equation describes a semicircle. Since we are taking x between 0 and 1, we will be finding one-half of that semicircle. If we had taken $-1 \le x \le 1$, we would have a semicircle.)

The number of points (x,y) such that $y < \sqrt{1 - x^2}$, divided by the total number of points, times the area of the square, equals the area of the quarter circle (it should be $\frac{\pi}{4} * r^2$, $r = 1$, therefore AREA $= \frac{\pi}{4} = .785$).

Let's see what the computer gets:

5 T←A←Y←N←M←0 (or--5 VAR=ZERO)	Multiple assignment setting T, A, Y, N, M all = 0.
10 LET N=500	Number of points we're trying.
20 FOR I=1 TO N 30 LET X=NUM(N)/N	Generates 500 possible values of x for $0 \le x \le 1$.
40 LET Y=NUM(N)/N	Same for y.
50 IF Y<SQRT(1-X↑2) LET T=T+1	If point is within quarter circle, increase counter by one.
60 LET M=M+1	Counts total number of points
70 NEXT I	
80 LET A=(T/M)*1 90 PR. "FOR ":I:" POINTS, AREA=":A 100 END	Area = points within the quarter circle/total points, times the area of the square.
>RUN	
FOR 500 POINTS, AREA=.756	The more points (up to a limit) the more accurate the area.

Caution:

Pay particular attention to the range of x and y. For instance, can you see why steps 30 and 40 in the above program would be meaningless for a circle of radius 2? The computer would respond: FOR 500 POINTS, AREA=1.000 Why?

Because the rectangle is totally within the circle, so the computer is really calculating the area of a square of side 1 which happens to be = 1.000. For a circle of radius 2, the range for x and y must be [0,2].

Hint:

A useful formula which helps to set the range on the random numbers generated is: FOR A<B, (B-A)*R+A , where A and B are the needed limits and R is a random number generator, generating values between 0 and 1. For instance, we want the range for a certain calculation to be 7 and 11. The formula we use is: x = $(11-7)*\underbrace{\text{NUM(N)/N}}_{R}+7$. For R = 0, x = 7; for R = 1, x = 11.

Your Turn:

1. Use 1,000 points to obtain a better approximation of the area of the quarter circle.

2. Modify the program on page 2 so that the user can input any radius. Don't forget the limits of x and y! Also, Have the program give the area of the entire circle.
3. Find the area under the curve $y = x$ for x between 0 and 10. Use an appropriate set of limits for y.
4. Find the area under the curve $y = \sin(x)$ for x between 0 and π.
5. Find the area under the curve $y = x^3 + 2x^2 + 3$ for $0 \leq x \leq 1$. Use x between 0 and 1, and y between 0 and 6. Then ask your teacher to use integral calculus to get an exact area under the same curve. Compare the two values.
6. Now show your teacher that you can easily find the area under the curve: $y = x^2\sqrt{(x^2 + 3^2)^3}$, for $0 \leq x \leq 1$, while he may find it not quite so easily. Hint to the student: Use x between 0 and 1, and y between 0 and 31.62278.

   Hint to the teacher: Use 0 and 1 as the limits of integration.

$$\int x^2\sqrt{(x^2+a^2)^3}dx = \frac{x}{6}\sqrt{(x^2+a^2)^5} - \frac{a^2x}{24}\sqrt{(x^2+a^2)^3}$$
$$- \frac{a^4x}{16}\sqrt{x^2+a^2} - \frac{a^6}{16}\log(x^2+\sqrt{x^2+a^2})$$

   Good luck!
7. Using a given curve, plot your accuracy versus the number of points used in the computer run. For instance, use the circle program on page 2. Try 10 points, 100 points, 200 points, 300 points, etc., and plot on a graph the increasing accuracy versus the number of points per trial.
8. See if you can think of some more interesting applications of the Monte Carlo method.

<u>Optional:</u>

1. Choose a very small area (such as .0001 by .0001) and see if the Monte Carlo method is any better for a small area.
2. Graph increasing area versus accuracy. (Note: accuracy should be calculated as the percentage difference between the Monte Carlo answer and the "correct" answer.)
3. Use the H-P Plotter to graph your curve, and then to plot each of the random points. Don't forget to also plot the rectangular region. You will get a very good idea of how Monte Carlo works by watching this process.

THE MONTE CARLO METHOD: TEACHER'S GUIDE

The main problem that students might have with this method is understanding how the ranges on x and y are set. What should be emphasized is that the maximum value of the ranges are the dimensions of the rectangle placed around the figure.

Another weak point is that students might not understand why the fraction T/M (the number of points in the figure/the total number of points tried) should be multiplied by the area of the rectangle. The fraction T/M represents the fraction of the rectangle taken up by the figure. This fraction of the rectangle multiplied by the area of the rectangle will give the area of the figure.

Students could be introduced to the integral calculus using this module. They should see that where the Monte Carlo method provides a very approximate estimate of the area under a curve, integration is defined to be the exact area under the curve because it takes the limit of an infinite number of dx's; similarly we could take an infinite number of points and come out with a precise value.

<u>Answers to 'Your Turn':</u>

1. Modify step 10 to read:
   LET N=1000

2. Modify program on page 2:
   Ranges: $0 \leq y \leq R$, $0 \leq x \leq R$

```
8 PR. "INPUT RADIUS":INPUT R
30 LET X=R*NUM(N)/N
40 LET Y=R*NUM(N)/N
50 IF Y SQRT(R 2-X 2) LET T=T+1
80 LET A=T/M*R 2
95 PR. "AREA OF ENTIRE CIRCLE=":4*A
```

   Might also include:

```
10 PR. "INPUT N":INPUT N
```

3. Modify program on page 2:
   Ranges: $0 \leq y \leq 10$, $0 \leq x \leq 10$

```
30 LET X=10*NUM(N)/N
40 LET Y=10*NUM(N)/N
50 IF Y<X LET T=T+1
80 LET A=T/M*100
   AREA=50
```

4. Modify program on page 2:
   Ranges: $0 \leq y \leq 1$, $0 \leq x \leq 3.14159265$

```
30 LET X=3.14159265*NUM(N)/N
50 IF Y<SIN(X) LET T=T+1
80 LET A=T/M*3.14159265
A  AREA=2.0
```

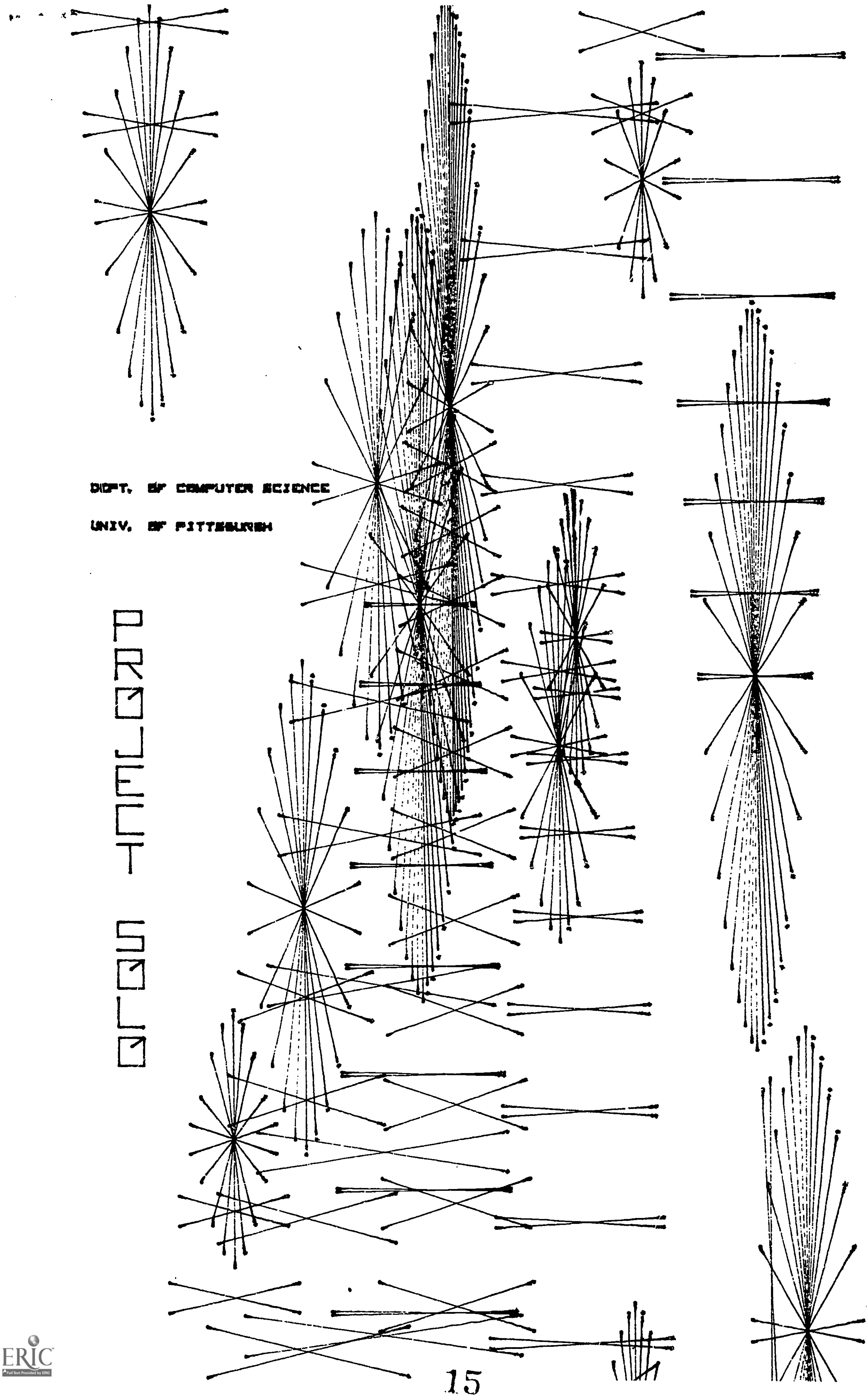
5. Modify program on page 2:
   Ranges: $0 \leq y \leq 6$, $0 \leq x \leq 1$

```
40 LET Y=6*NUM(N)/N
50 IF Y<(X↑3+2*X↑2+3) LET T=T+1
80 LET A=T/M*6.0
   AREA=3.91666  (by integration
                  and slide rule)
```

6. Modify program on page 2:

```
40 LET Y=31.62278*NUM(N)/N
   (31.62278=SQRT(1000))
50 IF Y<((X↑2)*SQRT((X↑2+9)↑3)) LET T=T+1
80 LET A=T/M*31.62278
   AREA=??????????
```

Note: Use a FOR-NEXT loop to automatically increase N in problem 7.


DEPT. OF COMPUTER SCIENCE
UNIV. OF PITTSBURGH
PROJECT SOLO

```
7 ØPEN // FØR ØUTPUT AS 3
10 K=0
12 B=NUM(100)
14 IF B<25 GØTØ 12
16 X=NUM(9999)
18 Y=NUM(8000)
19 IF Y<1000 GØTØ 18
20 Z=NUM(9999)
22 M=INT((X+Z)/2)
24 N=INT(ABS(X-Z)/2)
26 IF ABS(X-Z)<1000 ØR ABS(X-Z)>5000 GØTØ 16
30 PRINT ØN 3: "PLTP"
32 PRINT ØN 3: X;Y
34 PRINT ØN 3: "PLTL"
36 PRINT ØN 3: Z;Y
38 PRINT ØN 3: "PLTP"
40 FØR I=B BY B
41 IF I>M ØR Z-I<0 ØR X-I<0 ØR Y-I<0 ØR Z+I>9999 ØR X+I>9999 ØR Y+I>99
99 GØTØ 62
42 IF X<Z, X=X+I ELSE X=X-I
43 IF X<Z, Z=Z-I ELSE Z=Z+I
44 PRINT ØN 3: X;Y+I
45 PRINT ØN 3: "PLTL"
46 PRINT ØN 3: Z;Y-I
47 PRINT ØN 3: "PLTP"
48 PRINT ØN 3: Z;Y+I
49 PRINT ØN 3: "PLTL"
50 PRINT ØN 3: X;Y-I
51 PRINT ØN 3: "PLTP"
52 NEXT I
62 L=L+1, [IF L>3, GØTØ 66]
63 TYPE "ØNE DØNE"
64 GØTØ 12
66 PRINT ØN 3: "PLTT"
67 K=K+1, L=0, [IF K>3, GØTØ 76]
74 GØTØ 12
76 CLØSE 3
78 END
>RUN
ØPENING FILE 3, SYMBØLIC ØUTPUT, NAME: /STARS/
NEW FILE?
ØNE DØNE
ØNE DØNE
ØNE DØNE
ØNE DØNE
ØNE DØNE
ØNE DØNE
ØNE DØNE
ØNE DØNE
ØNE DØNE
ØNE DØNE
ØNE DØNE
ØNE DØNE

>-CØPY
FILE NAME:/STARS/
 TØ: TPT
PLTP
 7047      3483
  ....... etc. .......
```

Listing of a plotter program written by Chris Van Sickle (Grade 8). Part of the pattern produced on the plotter is reproduced on the cover.