Computer Science 120
Course Outline

#1.

I. Any questions, comments, etc.?
II. Discuss any difficulties or questions about last test.
III. Review grading procedure and W policy
IV. MAT functions --
   A. DIM statement (determines space limit; includes zeroth element or row zero and column zero which are NOT used by MAT functions.)
   B. MAT READ

```
10 DIM  D(10,20)     !limit of 11 x 21 = 231 elements
20 MAT READ D        !reads from DATA into the array, row by row
```

MAT READ D(7,12) would change the size and shape of D but not the maximum number of elements --one subscript might even exceed the original limits.

   C. MAT PRINT
      punctuation: none, comma, semicolon

```
100 MAT PRINT G;
```

   D. MAT INPUT

```
200 MAT INPUT A$     !note the possibility of a string matrix
```

   E. ZER, CON, IDN

```
100 MAT A = CON      !sets elements of A each equal to one
120 MAT B = ZER(5)   !to redimension B and zero the array
130 MAT C = IDN      ! sets elements on principal diagonal to 1, others to 0
                     ! thes[illegible]
```

[illegible]

1500 MAT A = B * C   ! Performs Matrix Multiplication

[illegible]

2

Computer Science 120
Course Outline

Day # 1. (cont)

V. System accounts and record keeping
   A. PPN and password security; usage category added at Wofford
   B. Usage records and the $MONEY program
      1. connect time in minutes
      2. CPU time in tenths of seconds
      3. device time ---usually plotter
      4. disk storage blokks (1 block = 512 characters)
      5. Disk storage limit (quota)
      6. $MONEY uses this informatinn from disk ---not inclusive of current session at terminal.
      7. Kilocore tics (tenths of seconds of CPU time x K of core used)
   C. System date and time functions
      1. TIME(N) --- returns a number
         a. N = 0 for seconds since midnight
         b. N = 1 for CPU time of current session (tenths of seconds)
         c. N = 2 for connect time of current session (in minutes)
         d. N = 3 for kilo-core tices for the current session (or job)
         e. N = 4 for device time in minutes for the current session(or job)

      2. TIME$(N) --- returnns a string
         a. N=0 to return the current system time in a form like 11:57 AM
         b. N non-zero to return the time string corresponding to N minutes before midnight

      3. DATE$(N) --- returnns a string
         a. N = 0 for the current system date
         b. N non-zero returns a date string as follows.
            the last three digits of the integgr N tell the number of the day in the year
            the remaining digits (0,1, or 2 numbers) tell how many years since 1970.

         example: DATE$(9074) = 15-Mar-79 (note the lower case characters)

TRY to use as little time as possible. It's cheaper for a changing system

"Another Kind of UsAge I want to talk about."

Computer Science 120

Day #2.

I. Program Efficiency Hints

A. to save storage space in core

1. Combine statements on a line to save statement header space (Statement headers are 6 words long)

Certain statements require headers anyway

DATA, DEF, DIM, FNEND, FOR, NEXT, and REM

use ! within statements lines for your comments and save the overhead

2. reduce the requirements for constant storage by requiring that each value be stored only once.

Consider the progressive improvements in program code:

```
10  A% = 278%
20  B% = 278%
```

vs.

```
10  A% = 288% : B% = 278%
```

vs.

```
10  A% = 278% : B% = A%
```

vs.

```
10  A%,B% = 278%
```

3. reduce the need for subscripted addressing (this also saves time)

```
10   For I% = 1% to N%
       : T = X(I%)
       : S = S + T
       : S2 = S2 + T*T
20 NEXT I%
```

4. eliminate unnecessary variables

```
10 A = B+C : D = A+E : F = D + G
```

becomes

```
10 F = B+C + E + G
```

if the intermediate values A and D are not needed elsewhere

5. use integer variables whenever possible (use % sign) as subscripts, many FOR ... NEXT loops, etc.

6. Use variables with the same first character as much as you can don't introduce a new first character unnecessarily

7. Re-use variables when you can. Make them do double duty --but you pay the penalty of making it more probable that you will make errors in your program logic.

B. to save computer time; writing faster code

1. use subroutines instead of user-defined functions
2. implied loops are faster than FOR ... NEXT loops (saves space also)
3. multiple IF statements might be replaced by compound IF statements or by ON .... GO TO .... statements

```
80 ON INSTR(1%,"XKBY", A$) GO TO 100, 200, 300, 400
```

will go to 300 if A$ is B

Day #2. (cont)

II. Ideas for homwgork.

A. Write a user-defined function which is the inverse of DATE$() -- i.e. which will give the date for any integer from 1 to 29365. Test your results by

FND(DATE$(N%)) = N% ???

A more ambitious project is to make one which will work from 1 to 32365. Note how DATE$ works for years after 1999.

B. Compare multiplication with exponentiation -- which is faster???

X↑N or X*X*X*X.....*X

or N = 2? for N = 3? FOR N = 6,7,8 etc. Is there a crossover point at which it is better to change your code??

C. Write a program which is the inverse of the TIME$() function.

D. Write a program to find the number of minutes and seconds since the beginning of the year.

Day #3. Information STorage Formats.

I. Binary number system
 A. used in computers because of its simplicity
 B. 2^n different symbols can be represented with n zeroes and ones
 C. one binary digit = 1 bit of information

II. Byte. (8 bits)
 A. $2^8 = 256$ different things to be represented with one byte.
 B. ASCII character code, appendix E.1
  allows for 256 different characters (plenty!)
 C. each character of a string is stored as one byte
  Example: the character ; is 59_{10} or 00111011_2 in the ASCII code

III. Word.
 A. In the PDP 11/40, 2 bytes or 16 bits make one computer word.
 B. Integers are stored in one word on our system
  1. the bits are numbered from 0 thru 15
  2. bit # 15 is the 'sign bit' : 0 for + and 1 for -
  3. bits 0-14 contain the value of the integer in binary notation
  4. limit on size:
   $2^{15} = 32768$ things (the numbers from 0 thru 32767)

  5. Complement form for negative numbers. (twos complement)

   -3 is stored as the two's complement of 3 (3 = 0000000000000011)

   -3 = 1111111111111100 + 1 = 1111111111111101
   -1 = 1111111111111111
  6. smallest integer is -32768.
   -0 is not needed, so 1000000000000000 is used as -32768

   !!largest integer - smallest integer

   0111111111111111
   1
   1000000000000000

   also 32767 + 2 = -32767 etc..

IV. Two word storage of floating point numbers.
 1. first word
  a. bit 15 is again the 'sign bit' as before
  b. bits 7-14 contain the binary exponent in excess 128 notation
  c. bits 0-6 contain the high order (most significant part of the) mantissa without 1st bit
 2. second word
  a. bits 0-15 contain the low order digits of the mantissa

 3. effectively 24 bits of mantissa; $2^{24} = 16,777,216$ or 7+ decimal digits
 4. the power of 2 can be from -128 to 127
  $2^{-128} = 2.93873588 \times 10^{-39}$
  $2^{127} = 1.70141183 \times 10^{38}$

Computer Science 120

Day #3. (cont)

5. sample storage for floating point numbers

$1385_{10} = 10101001110_2 = .1010100111 * 2^{11}$

since 1385 is positive, the sign bit is 0
add 128 to the exponent: $139_{10} = 10001011_2$ (exponent bits)

omitting the first bit after the binary point, insert the mantissa of the number ('hidden bit')

word 1: 0100010110101001 (or as 2 bytes) 01000101 10101001

word 2: 1100000000000000 (or as 2 bytes 11000000 00000000

V. Computer numbers

1. limited precision

2. not a continuous set --and the gaps are of different sizes in different parts of the number system.

as seen above, there is no number between 0 and about 1.4 E-39

VI. functions which convert the storage format from one to another

1. Y% = ASCII(A$) returns the decimal value of the first character in A$, using the ASCII code
2. Y$ = CHR$(X%) returns the string character represented by the value of X%, using the ASCII code

3. CVT%$, CVTF$, CVT$%, CVT$F

NB: each of these functions swaps the order of the bytes
the F functions swap the word order also so that a number in floating point format in bytes 1,2,3,4 is converted to a string with the bytes in 4,3,2,1 order.

4. CVT$$(A$,N%) --see table in DEC manual for significance of various values of N%. VAlues may be added for combined effects.

5. Y$= NUM$(A) produces the string that would be printed by PRINT A

6. Y = VAL(A$) if A$ contains the characters that form a valid number.

7. Y%= SWAP%(N%) interchanges the two bytes of N%, giving new integer Y%

8. CHANGE A$ to X% where X%is a list and will contain the ASCII values of the characters of A$. X%(0) is the length of A$.
9. CHANGE X%to A$

10. STRING(L%,C%) creates a string of L% characters long where each is the character represented by C% in ASCII code.

VII. Write a program to determine and print each bit of the representation of the floating point number 678.456.

Computer Science 120
Course Outline

Day #4. Logic

I. Integer variables as logical variables
   A. truth values : T= -1%; F = 0%; any non-zero value will be True
   B. Program switches --set somewhere in program and test later

```
A% = (p$ ="stop")
B% = (X> 12)
C% = D% and 4%
IF A% and (B% OR C%) then 1280
```

II. bit manipulation --using less than full word for data storage

See truth tables in DEC manual

A% = 0110 (data word) of course there are 16 bits in our
B% = 1010 (mask word) computer word --Ishortened these for simplici

function	result	comment
A% AND B%	0010	logical product. copy selected bits of data work & set others to 0. can be used to copy part of a work, to see if certain bits are set or set certain bits to 0.
A% OR B%	1110	to set selected bits to 1 and copy all others. logical sum.
A% XOR B%	1100	logical difference --shows, bit by bit, where A and B differ.
A% EQV B%	0011	shows, bit by bit, where A and B match
A% IMP B%	1011	the order of A and B matters here sets all bits of C except here A has a 1 and B has a 0.

Examples:

If (A% AND NOT 7%) = 0	same as IF A%<7% ??
IF (NOT A% and 7%) = 0	if last three bits of A are 1s
C% = A% AND 240%	copy bits 4-7 of A into C; perhaps this part of the word represents data we want.
C% = A% or 12%	sets bits 2 & 3 to 1s, copies all others
C% = A% ornot 12%	copies bits 2&3 and sets all others to ones.

needed operation s:

1. copy word segment	C% = A% and 240%
2. set certain bits to 1 without changing others	A% = A% or 12%
3. set certain bits to 0 without changing others	A% = A% AND NOT 12%
4. test bit N	IF (A% and 2%↑N%) = 2%↑N% Then (or test for 0)
5. set bit N	A% = A% and 2%↑ N%
6. turn off bit N	A% = A% and NOT 2%↑N%

Computer Science 120
Course Outline

Day #5. Random numbers.

I. Pseudo-random number generation *(one way)*

A. $R_n = C*R_{n-1} \pmod N$ *($N=5$, $C=2$, $R_0=1$ → 1,2,4,3,1,2,4,3,.... choose large N (32768))*

B. concept of [illegible]tion length

1. define *(Choose C≠0 ~~8n±3~~, ~√N /32767; Choose R_0 = odd)*
2. depends on choice of N, C and R_0
   a. repetition length increases with N, so choose $N = 2^{b-1}$ (32768)
   b. choose C of the form 8n±3 where n is any positive integer
   c. choose R_0 to be an odd integer
   d. choose C nedr the square root of N for better statistical properties, although not inceased repetition length.

C. divide R$_n$ by N to get a number in the range 0 to 1

II. Statistical tests on random numbers (and random digits)

(To eliminate gap — CHANGE N every k number of times.)

1. mean test -- should be near 0.5 in the above case
2. Frequency distribution test -- equal intervals should contain equal counts of randomly generated values.
3. frequency of runs test. A run of length k is a series of k consecutive numbers which is monotonic (either increasing or decreasing)

   there should be $\frac{2}{(k+3)!}\left[N(k^2+3k+1)-(k^3+3k^2-k-4)\right]$

   runs of length k in a series of N random numbers

4. runs above and below the mean test
   ---look for sequences which are either all above or all below the mean.
   there should be $\frac{(N-k+3)}{2^{k+1}}$

   such runs of length k in N numbers.

5. gap test for random digits
   each digit should recur, in a sequence, every 10 times (on the average)
   i.e the average "gap" between repetitions of a digit should be 10.
   This should be true for each of the digits

   $\text{Variance} = \frac{\sum_{i=1}^{N}(g_i-\bar{g})^2}{N}$; g_i = current gap; $\bar{g}$ = expected mean

   VARIANCE = average value of the squared deviations from the mean.
   For random digits, the VARIANCE of the gaps (as above) should be 90

6. The maximum test
   In a set of three random digits, the center one will exceed the other two in 285 out of 1000 cases.

   gap=3 (123)

III. Uses of Random numbers.

1. Simulation
   representing portions of a non-deterministic process or of one in which the deterministic rules are unknown or difficult to apply
2. Monte Carlo techniques applied to definite integrals
   area, volume, center of mass, etc.

Possible program assignments:

1. Write your own pseudo-random number generator and select poor values (then good values) of R_0, C and N and find the repetition length in each of several cases.

✓ & see back.

Computer Science 120
Course Outline

2. Write a program to apply the MAXIMUM test to the random number generator usdd by the BASIC-PLUS RND function.
3. Write a program to find what volume is left when a sphere of radius 2 has a hole of radium 1 drilled thru it along a diameter. Use the Monte Carlo technique.
4. Write a program to find the mass of a sphere of radius R_o =2 and density

$$1 + e^{-R/R_o}$$

5. Consider a set of 1000 radioactive nuclei. Each has a .02 probability of decay in one unit of time. How may decay in each of the first 20 units of time and how many remain at each seep? Give each undecayed nucleus a chance to decay in each new time integval.

Day #6. Sorting

I. define problem, note importance
II. references, etc.
   A. !HEPSRT
   B. KNUTH --The Art of Computer Programming
   C. Creative Computing, vol #2 issue #6
III. bubble sort

```
500  FOR I = 1 TO N
510     FOR J = I+1 TO N
520        IF D(I)< = D(J) THEN 560
530        T = D(J)
540        D(I)= D(J)
550        D(I) = T
560     NEXT J
570  NEXT I
```

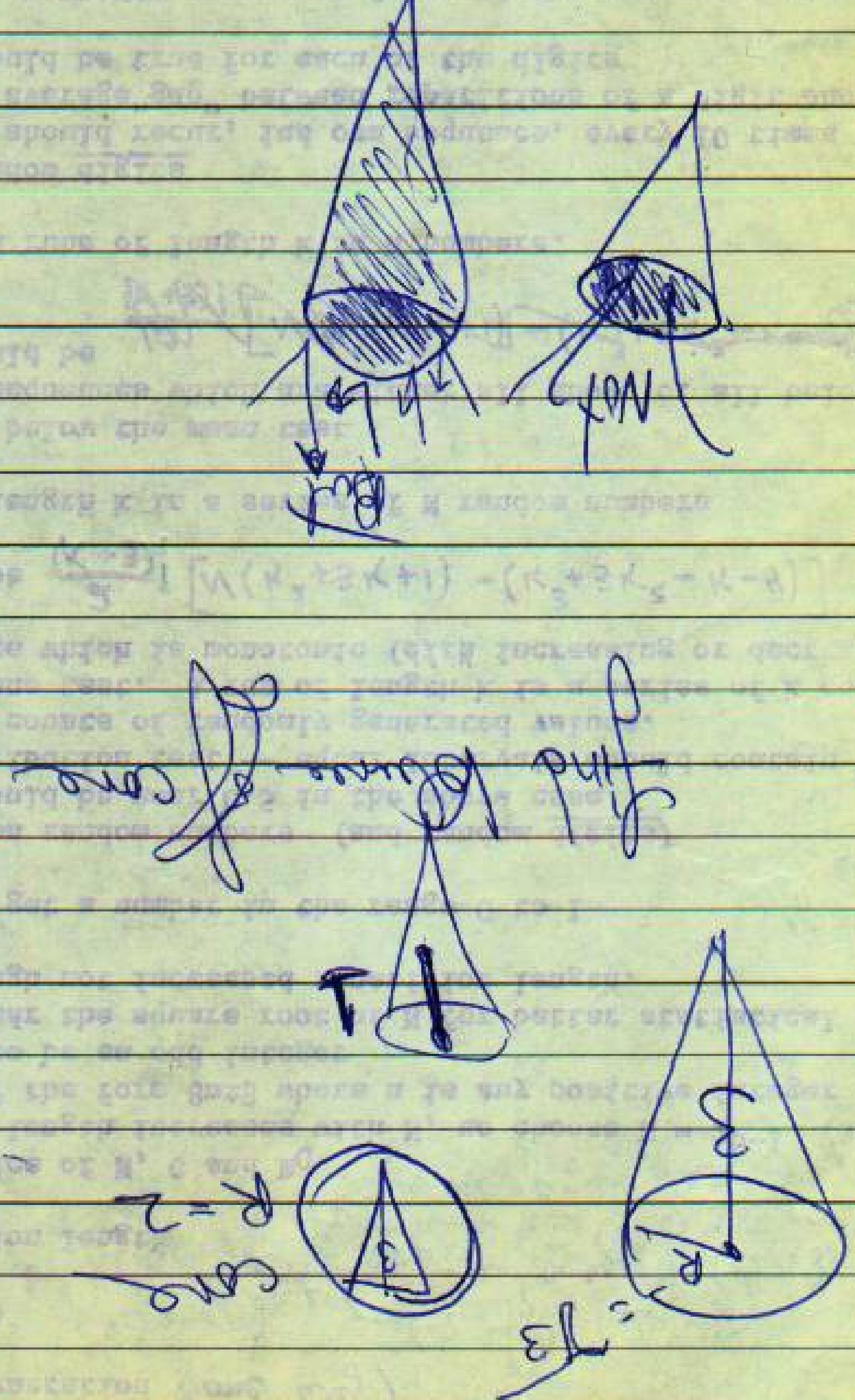
Program segment to sort a list D of N items into increasing order.

IV. delayed replacement sort

```
500  FOR K = 1 TO N
510     I = K
520     FOR J = I+1 TO N
530        IF D(I)< = D(J) THEN I = J
540     NEXT J
550     IF K = I THEN 600
560     T = D(I)
570     D(I) = D(K)
580     D(K) = T
590  NEXT K
```

V. Shell-Metzner sort (Shell sort modified by Marlene Metzner)

```
READ Y1Y 700H  JPLY JSL

700 FORMAT Y2I10H

GO TO 7

END

18

11 575C LNKEDT

11 575C
```



Find from Int 1-200
those in which Bits 2, 4 & 6
are off but #5 is on

& Int
Switch on/off states of 5 & 6
& Int (New)
Copy 4 low-order Bits of ~~and~~ original #.