Sec: 1 2 (circle) ECE-1021 Fall 2004 Exam #1 07 October 2004

NAME: KEY Score:_____/100

Unless specified otherwise, assume that:

- The variables i, j, k, l, m, and n are declared as int
- The variables u, v, w, x, y, and z are declared as doubles.
- Any reference to a "floating point" value is to be taken as a type double.
- Any reference to an "integer" value is to be taken as a type int.

Multiple Choice (2 points each) – Pick the SINGLE best answer for each question.

1) Which of the following are complementary pairs of operators?

a) (= = , !=), (>=,<), (>, <=)
b) (= = , !=), (>,<), (>=, <=)
c) (= = , !=), (>,<), (>=, =<)
d) (= = , !=), (>,>=), (<=, <=)

2) Which of the following assignments produces a value of zero?

a) result = 2 – 6%3; 6%3=0
b) result = 2 – 8%3; 8%3=2 ✓
c) result = 9%3 – 1; 9%3=0
d) result = 8%3 – 1; 8%3=2

3) How many values can 16-bits represent?

a) 256
b) 32,768 $2^{16}=65536$
c) 65,536
d) 1,048,576

4) Which of the following statements is false?

a) A logical operator will always return a value of 0 or 1. T
b) A value is a logical TRUE if it is equal to –1. T
c) A value is a logical FALSE if it is equal to 0. T
d) Negative values cannot be interpreted as logical values. F (THE ARE LOGICAL 'TRUE')

5) Modulo division of m by n (m%n) is used to return

a) The same value that would be returned by m – n*(m/n).
b) The result of an integer division. F
c) A value of 1 if m can be evenly divided by n. F (WOULD BE 0)
d) The result of a floating point division even if the operands are integers. F

6) What must be true of an else statement?

a) It must contain a logical test.
b) It is associated with the nearest if() statement above it.
c) It must be properly indented for the compiler to determine which if() statement is controlling it.
d) It must appear immediately after an if() statement in the program structure.

7) Consider the following statement: int i = 100, j = 0; Which of the following statements is true?

a) (i>0) || (j>50) (i>0 T) j>50 F
b) i < 3 F
c) !(j<1) (j<1)T !T→5
d) (j<1) && (i<=10)
T F

8) Why does C offer three different looping structures?

a) Because no one looping structure can implement all of the different types of loop logic that might be needed by a program.
b) Because the for() loop is only capable of executing a finite and predetermined number of times and the while() loop is not guaranteed to execute the loop code at least once.
c) For compatibility with other languages.
d) Because each structure lends itself to a certain type of looping logic and program readability and maintainability are enhanced if the structure used matches the logic implemented.

9) What is the purpose of indenting various lines of code different amounts?

a) It is a requirement of the language – the amount of indenting is used by the compiler to determine code structure.
b) It is purely for aesthetic value and is completely arbitrary. There is no significant advantage to indented code versus non-indented code.
c) The indenting of certain functions, such as loops and if()...else structures, are mandatory and the rest is just to make the appearance of the code more consistent.
d) It is purely to aid the programmer in determining code structure quickly and accurately – the compiler ignores all indenting.

10) What is the value of the following expression?

```
!( (25 - 55%10) < 20 || (24/10 > 2) )
```

5
20 < 20
F
2 > 2
F
T F
T

a) true.
b) false.
c) invalid.
d) none of the above.

11) What are the three basic building-block structures of a structured program?

a) Input statements, output statements, and computation statements.
b) Sequences, selections, and repetitions.
c) Goto's, loops, I/O.
d) while() loops, for() loops, and do/while() loops.

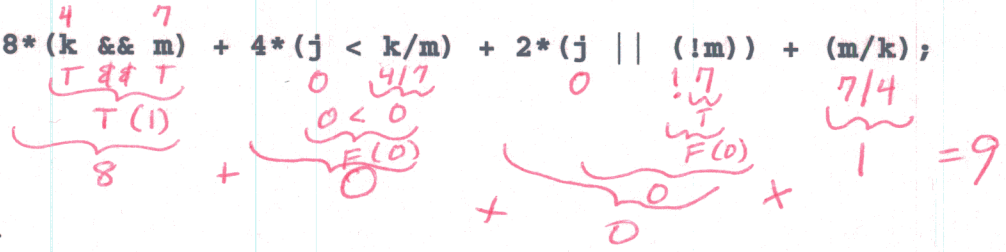
12) What are the two common text-based ways of indicating that a number is written in base-16?

a) Prefixing the number with "0x" or placing an "H" as a suffix.
b) Suffixing the number with "x" or placing an "H" as a prefix. F
c) Placing "/16" after the number or placing "(16x)" before the number. F
d) Placing "base-16" to the left of the number or, alternately, to the right of the number. F

13) If j = 0, k = 4 and m = 7, what value is stored in n?

```
n = 8*(k && m) + 4*(j < k/m) + 2*(j || (!m)) + (m/k);
```

a) 8.
b) 9.
c) 9.75.
d) 15.75.



14) What is required for a (nonzero) number to be in a normalized exponential format?

a) The mantissa has exactly one non-zero digit to the left of the radix point.
b) The exponent is the smallest possible value it can be.
c) The mantissa is an integer.
d) The exponent is the largest possible value it can be.

15) What is the defining relationship for the concept of a negative number?

a) A positive number with a minus sign in front.
b) A number and its negative will sum to zero.
c) A number and its negative have the same absolute value.
d) The negative of a number is less than zero with the same absolute value as the number itself.

16) Which of the following hexadecimal digits is a prime number?

a) A = 10
b) C = 12
c) D = 13 ✓
d) F = 15

17) Why are octal and hexadecimal so prevalent in computer science?

a) Because binary bit strings can be converted to and from octal and hexadecimal very quickly. ✓
b) Because the first programmers set the standard and we are stuck with it.
c) Because arithmetic is easier in either of these two bases than it is in decimal.
d) Because learning hexadecimal serves as a "right of passage" for would be programmers.

18) What is the most likely reason that a "nibble" is the size that it is?

a) Because that is exactly half a byte.
b) Because the first processor has a data bus that was this size.
c) Because that is the smallest number of bits that can be used to represent all of the decimal digits.
d) Because anything larger would require the use of a number base that is too difficult for humans to work with.

19) Which of the following is not one of the allowed ways to represent negative integers under the C Language Standard?

a) Offset binary.
b) Signed binary.
c) One's complement.
d) Two's complement

} ALL THREE USE IDENTICAL REPRESENTATIONS FOR POSITIVE VALUES (AND ZERO)

20) What property does two's compliment exploit?

a) The fact that, in a fixed-width binary value consisting of N bits, that 2^N is indistinguishable from zero.
b) The inverting all of the bits in a value produces a one's compliment value.
c) That subtraction is equivalent to adding the negative of a number.
d) That addition and subtraction use the same hardware in a computer.

21) How many bits are used by the standard ASCII code?

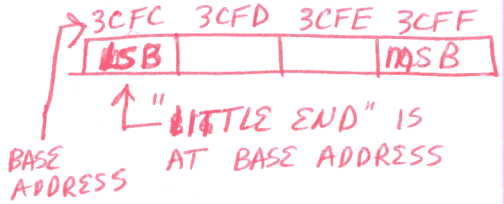
a) 4.
b) 7.
c) 8.
d) 16.

22) How is the end of a string of characters indicated in many programming languages, including C?

a) By following the string with an additional byte that is set to all zeroes - the NUL character.
b) By interpreting the first byte in the string as an integer that indicates how many characters are in the string.
c) By passing the length of the string as a separate argument to any function needing that information.
d) By marking the end of the string with the newline ('\n') character.

23) A four-byte integer is stored in addresses 0x3CFC through 0x3CFF with the most significant byte at location 0x3CFF. What is true about this representation?

a) It is stored at 0x3CFF in Big Endian format.
b) It is stored at 0x3CFF in Little Endian format.
c) It is stored at 0x3CFC in Big Endian format.
d) It is stored at 0x3CFC in Little Endian format.



24) All ASCII characters fall into exactly one of which of the following pairs of groups?

a) Printing and Control.
b) Upper case and Lower case. NUMBER?
c) Alphanumeric and Punctuation. CONTROL?
d) Control and Graphical. SPACE?

25) What ASCII codes would be sent to the screen to print the value of pi, namely 3.14?

a) 3.14.
b) 0x03, 0x2E, 0x01, 0x04
c) 0x33, 0x2E, 0x31, 0x34
d) 0x70, 0x69

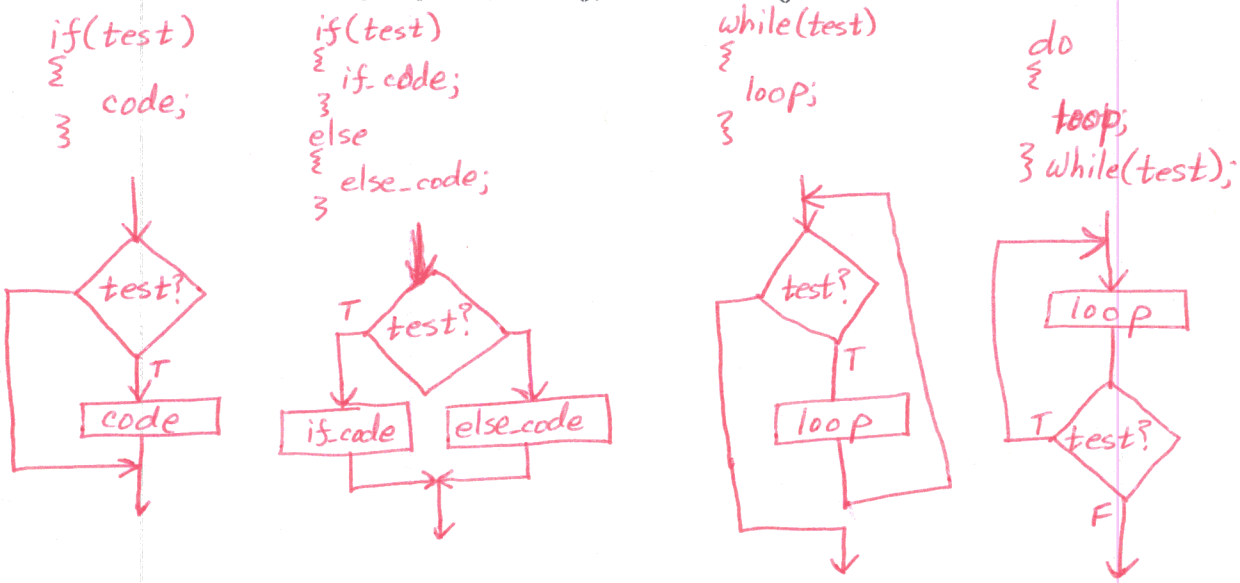
'3' = 0x33
'.' = 0x2E
'1' = 0x31
'4' = 0x34

Short Answer (4pts each)

26) The string, "Fall 2004" is stored at memory location 0xC340. Indicate the values (using the actual numbers - in hex) that are stored in each of the relevant memory locations.

C340	41	42	43	44	45	46	47	48	C349
46	61	6C	6C	20	32	30	30	34	00
F	a	l	l	SP	2	0	0	4	\0

27) Draw the flow charts for an if(), if()/else, while(), and do/while() constructs.



28) What value is stored in k?

```
k = 555/10*10;
```

55

550

550

29) If a (non-global) variable is not initialized, what value will it have by default?

garbage - whatever was already in memory at that location.

30) Briefly summarize the key elements of the top-down algorithm development strategy.

DIVIDE-AND-CONQUER

BREAK THE PROBLEM INTO SMALLER TASKS.
TREAT EACH TASK AS A SEPARATE PROBLEM.
INTEGRATE TASK SOLUTIONS INTO OVERALL PROBLEM SOLUTION

Longer Answer (10 pts each)

General Note: You must supply the code for any other function(s) or macro(s) you wish to call as part of the code for the function you are asked to write. Be sure to indicate which, if any, header files must be included to make your function work.

31) Write a function, called PutHex(), that outputs a single hexadecimal digit to the screen. The argument to the function is a single integer value that is expected to be between 0 and 15, inclusive. For values less than 10, the normal decimal digit should be displayed. For values between 10 and 15, the function should display an uppercase 'A' through 'F', respectively. Values outside of this range should print a period to the screen. The function returns a 1 if the output is a valid hex digit and a 0 if it is not.

```
#include <stdio.h>

int PutHex(int d)
{
    if (d<0)
    {
        putc('.', stdout);
        return 0;
    }
    if (d<10)
    {
        putc('0'+d, stdout);
        return 1;
    }
    if (d<16)
    {
        putc('A'+(d-10), stdout);
        return 1;
    }
    putc('.', stdout);
    return 0;
}
```

32) Write the function:

char *strupr(char *s);

This function converts every lower case character in the null-terminated string pointed to by s to the corresponding upper case character. All other characters are left unchanged. The function returns a pointer to the string passed to it. You must supply the code for any other function(s) you wish to call as part of the code for this function.

```
char *strupr(char *s)
{
   int i;
   i=0;
   while('\0' != s[i])
   {
      if( ('a' <= s[i]) && (s[i] <= 'z') )
         s[i] += 'A' - 'a';
      i++;
   }
   return s;
}
```

or

```
int isupper(int c)
{
   return( ('a' <= c) && (c <= 'z') );
}
int toupper(int c)
{
   return( isupper(c)? c+'A'-'a' : c );
}
char *strupr(char *s)
{
   int i;
   for(i=0; '\0' != s[i]; i++)
      s[i] = toupper(s[i]);
   return(s);
}
```

33) The function PutS() that we have performs a very specific task – it takes as an argument the address where a string of text is stored in memory, it displays that text on the screen using repeated calls to the putc() function (via the PutC() macro), and it returns the total number of characters output to the screen.

Our version of PutS() is based on the premise that the string whose address is passed is a null-terminated string. This is not the only way that the string could have been stored. The other common way of representing strings is to prefix them with a character count. In this case, the function would still be passed a pointer to a one-byte integer (i.e., a char) but the bits at that location would be interpreted as the number, N, of characters in the string. The characters themselves are then located in the next N bytes of memory.

For instance, if the string "Hello World!" were stored at memory location 0x342 under each convention, the memory map in that region might look like:

null-terminated string:

340	341	342	343	344	345	346	347	348	349	34A	34B	34C	34D	34E	34F
68	91	48	65	6C	6C	6F	20	57	6F	72	6C	64	21	00	09
'h'	.	'H'	'e'	'l'	'l'	'o'	SP	'W'	'o'	'r'	'l'	'd'	'!'	NUL	HT

count-prefixed string:

340	341	342	343	344	345	346	347	348	349	34A	34B	34C	34D	34E	34F
68	91	0C	48	65	6C	6C	6F	20	57	6F	72	6C	64	21	09
'h'	.	FF	'H'	'e'	'l'	'l'	'o'	SP	'W'	'o'	'r'	'l'	'd'	'!'	HT

Write a function called PutSP() that performs the same task as PutS(), but for count-prefixed strings.

```
#include <stdio.h>
int PutSP(char *s)
{
   int i;
   i=0
   while ( i< s[0])
   {
      putc( ~~stdio~~ s[i+1], stdout);
      i++;
   }
   return s[0];
}
```

EXTRA CREDIT – (10 pts) In accounting, it is common practice to indicate negative values by surrounding the absolute value of the number with either parentheses or angled brackets. Write a function called PutV_lfa() that takes a single argument of type double and displays the value to two decimal digits (properly rounded) with a dollar sign preceding the value. If the value is negative, then angle brackets surround everything. The value returned should be the total number of characters printed.

Example 1: PutV_lfa(3.4176); /* Should display: $3.42 Should return 5 */
Example 2: PutV_lfa(-15.7323); /* Should display: <$15.73> Should return 8 */

```
TASK #1: IF (x<0) OUTPUT '<'
TASK #2: OUTPUT '$'
TASK #3: SET y = |x|
TASK #4: ROUND y to nearest cent
TASK #5: OUTPUT y to 2 PLACES
TASK #6: if (x<0) OUTPUT '>'
```

```
#include <stdio.h> /* putc() */
#define PutC(c) (putc((char)(c), stdout))
#define PutD(d) (PutC((d)+'0'))
int PutV_lfa(double x)
{
   int count, i;
   double v, wt;
   count=0;
   v= (x>=0.0)? x : -x;
   if (x<0.0)
   {
      PutC('<');
      count++;
   }
   PutC('$');
   count++;
   v+=0.005;
   for(wt=1.0; v/wt >=10.0; wt*=10.0);
   while (wt>0.5)
   {
     for (i=0; v>=wt; i++)
        v-=wt;
     PutD(i);
     count++;
     wt /= 10.0;
   }
   PutC('.');
   count++;
   while (wt>0.005)
   {
     for(i=0; v>=wt; i++)
        v-=wt;
     PutD(i);
     count++;
     wt /= 10.0;
   }
   if (x<0.0)
   {
      PutC('>');
      count++;
   }
   return count;
}
```