THE UNIVERSITY COLLEGE OF THE CARIBOO

MIDTERM EXAMINATION

COURSE: **COMP 213** **INSTRUCTOR:** Yanni Giftakis
SEMESTER: Fall 2004 **DATE:** October 22, 2004

Total: / **33** marks

Part 1 – General Multiple Choice – 15 Marks [1 mark each]

____ ***1. The acronym* CPU *means,***

a. cortical plumbing upset
b. cheese-painted underwear
c. Canadian public university
d. central processing unit

____ ***2. The programming language that uses hexadecimal to represent 0's and 1's.***

a. machine language
b. microcode
c. assembly language
d. C

____ ***3. CMOS is _____.***

a. the cache in the CPU
b. one of the ROM chips
c. a type of memory
d. a processor in the BIOS

____ ***4. Overflow errors are related to which part of a floating-point representation?***

a. exponent
b. mantissa
c. the entire representation
d. overflow errors do not occur with FP

____ ***5. Which of the following Linux commands displays the amount of free diskspace?***

a. df
b. top
c. free
d. fdisk

____ ***6. Bit patterns stored in memory can be distinguished easily as integers, characters, or floating-point representations.***

a. true
b. false

____ ***7. 16 bits are used to describe characters in the Unicode character coding scheme. How many possible characters/symbols can be represented in Unicode?***

a. 128
b. 32,768
c. 256
d. 65,536

____ ***8. L1 cache can be found _____.***

a. on the mainboard
b. on the cpu die
c. either on the cpu die or mainboard
d. on the bus

____ ***9. What is "most significant" about the most significant bit?***

a. it stands out from the rest of the bits
b. it is the first bit
c. it is *always* the sign bit
d. it holds the largest binary unit value

____ ***10. What is the default executable file created by any GNU compiler in UNIX/Linux?***

a. run.exe
b. a.out
c. whatever you call it with the **–o** option
d. the name of the source file, with no extension

____ ***11. Overflow errors are detectable, whereas underflow errors are undetectable.***

a. true
b. false

____ ***12. In Linux, if you wish to access a partition, drive, or device in Linux, it must first be _____.***

a. formatted
b. iconised
c. mounted
d. loaded

____ ***13. Recalling the values of "true" and "false" for C logical expressions, what is the output?***

```
int value = 3;
( value ) ? printf("t") : printf("f");
```

a. t b. nothing
c. f d. invalid; value must be defined of type **boolean**

____ ***14. Which of the following is not an advantage of a RISC-based processor over CISC?***

a. executes program instructions quickly b. uses less energy per instruction cycle
c. allows for shorter programs d. smaller instruction set

____ ***15. The names of the creators of the ANSI standard C programming language are _____.***

a. Kaplan and Rasspi b. Kellogg and Rapani
c. Koridon and Ractagino **d. Kernighan and Ritchie**

Part 2 - Written Answer – 18 Marks

For questions: 1 to 4, use this 16-bit word: **0 1 1 1 0 1 0 1 1 1 0 0 0 1 0 0**

[1 mark]

1. Represent the word in hexadecimal: __ $_{16}$

   each hex digit represents 4 bits (30148_{10}),

   0 1 1 1 | 0 1 0 1 | 1 1 0 0 | 0 1 0 0 $_2$
   7 | 5 | C | 4 $_{16}$

[1 mark]

2. If the first byte on the left represents an 8-bit word, 2's complement, it represents what value? __ $_{10}$

   $0\ 1\ 1\ 1\ 0\ 1\ 0\ 1_2 = 2^6 + 2^5 + 2^4 + 2^2 + 2^0 = 117_{10}$

[2 marks]

3. What is the numeric range, if the word represents, (answer in both binary __ $_2$ and decimal __ $_{10}$),

   a) a 16-bit word, unsigned integer (0 to largest positive)

   largest positive: $1111\ 1111\ 1111\ 1111_2 = 65{,}535_{10}$
   zero: $0000\ 0000\ 0000\ 0000_2 = 0_{10}$

   b) a 16-bit word, 2's complement integer (largest negative to largest positive)

   largest positive: $0111\ 1111\ 1111\ 1111_2 = 32{,}767_{10}$
   zero: $0000\ 0000\ 0000\ 0000_2 = 0_{10}$
   largest negative: $1000\ 0000\ 0000\ 0000_2 = -32{,}768_{10}$

   FYI: if the example word was viewed as signed or unsigned = $30{,}148_{10}$

[2 marks]

4. If it represents floating-point with: 6-bit exponent, 9-bit mantissa; what is the true decimal value? __ $_{10}$

   0 | 111010 | 111000100

   1. sign bit: 0 - positive true value
   2. exponent: $111010_2 = -6_{10}$
   3. mantissa: $1.110001_2 * 2^{-6} = 0.000001110001_2$
   4. $= 2^{-6} + 2^{-7} + 2^{-8} + 2^{-12} = 0.02758789_{10}$

[1 mark]

5. What does the following Linux statement accomplish?

```
user@linux> ls *.c | sort > list.txt
```

the command obtains a **lis**ting of all files in the current directory with extension **.c**, passes the list to the **sort** command which order the list alphabetically, and the sorted output is sent to the file **list.txt**

[1 mark]

6. In Linux, identify two (2) methods for obtaining command help directly from the operating system; for example on the commands: **cp** and **passwd**

there are three (3) common methods for obtaining help from within the Linux operating system, the example below use the **cp** command as an example,

man cp or **info cp** or **cp --help**

[1 mark]

7. One of your classmates finished typing a large C program and is rather proud of the impressive data-structures and efficient code techniques. After compiling it successfully with no errors, the source file is opened in an editor. *Shockingly, the source file is now just a collection of strange ASCII characters!!*

The following commands had been typed,

```
user@linux> gcc myprogram.c -o myprogram.c
user@linux> ./myprogram.c
user@linux> jed myprogram.c
```

To save the sanity of your classmate, explain what happened?

your classmate was careless and sent the executable output from the **gcc** compiler to the same filename as the source code—this effectively *overwrote* the source code.
tell your classmate to be more careful when naming things.

[1 mark]

8. Write a C program to sum a series of floating-point values from the user (use **scanf()** and the format "%f"). The program continues asking for values until the number 0.0 is entered. It then displays the average of the values to the screen.

```
int main()
{
   double sum=0.0, average=0.0, val=0.0;
   int    count=0;

   printf("Enter the first value: ");
   scanf ("%f",&val);        // get first value

   while ( val != 0.0 )
   {
      count++;              // incr. value counter
      sum = sum + val;      // accum value
      printf("Next value: ");
      scanf ("%f",&val);        // get first value
   }
   ( count>0 ) ?  average = (sum / count) : average = 0;
   printf ("The average is %f \n",average);

}//end of main()
```

[1 mark]

9. After declaring the double pointer **p**, allocate a single memory location and assign the reference to p.
   a) using **malloc()** b) using **calloc()**

```
a) double *p = (double*)malloc( 1*sizeof(double) );   // 1* is optional
b) double *p = (double*)calloc( 1, sizeof(double) );
```

[2 marks]

10. Identify, and fix, all problems with the following program.

```
include (stdio.h)
int main ()

   int age;
   char firstname[];

   printf ("Enter you age and first name:");
   scanf ("%d %s", age, &firstname);

   printf ("\nHi %s, you are %d years old", age, firstname);

   return (0);
}
```

corrections are in **bold underline**,

```
include <stdio.h>
int main ()

   int age;
   char firstname[40];

   printf ("Enter you age and first name:");
   scanf ("%d %s", &age, ~~&~~firstname);   ← the array is already a pointer

   printf ("\nHi %s, you are %d years old", ~~age, firstname~~);
   printf ("\nHi %s, you are %d years old", firstname, age);

   return (0);
}
```

[2 marks]

11. Write a program that displays the exact number of bytes in the data types: **long**, **double**, and **char**.
    You are not permitted to use the function **sizeof()**
    (*hint*: the memory address difference between two variables of the same type)

    only the code segment is provided to determine the size of the **double** type,

```
double a, b;          // declare to double variables
int sod = 0.0;        // size of double

  // knowing that b will be declared earlier in memory than a
sod = (int)&a - (int)&b;  // numeric difference in addresses
```

[1 mark]

12. Complete the function **arraysize()**, that determines the number of elements in an array of type double that always has the value 0.0 as its last element (do not count this element).
The header of the function is provided.

```
int arraysize(double *list)
{
   int count = 0;      // number of values
   for (count = 0; list[count] != 0.0; count++);   // count number of values
   return (count);
}
```

[2 marks]

13. Complete the code segment that parses a string (character array) and displays each character on it's own line until the end (recall that the end of a string is the sentinel symbol: '\0').

You are not allowed to use the array **phrase[]**, nor any array notation to access elements (no [ ])
(*hint*: use pointer referencing/dereferencing).

```
char phrase[] = "The dog's name is Blue.";

char *p;      // declare pointer

for (p = phrase; *p != '\0'; p++)   // parse array until null ('\0')
   printf ("%c\n",*p);                // or:  putchar(*p); putchar('\n');
```

[2 marks] -- bonus

14. The **scanf()** function is wonderful for efficiently accepting almost any primitive data type, such as ints and doubles, and even strings (character arrays).

Write your own string input function **scanstring()** that returns nothing, and has a single parameter: an existing array of characters. You are not permitted to use the **scanf()** function.
(*hint*: use the **getchar()** function to get a single character; and recall the end of input/"enter" symbol: '\n')

easy way, using **gets(); but this function is not recommended

```
void scanstring (char *inputstr)    or    void scanstring (char inputstr[])
{                                         {
   gets (inputstr);
}
```

expected way, using **getchar()

```
void scanstring (char *inputstr)    or    void scanstring (char inputstr[])
{                                         {
   char inp = ' ';       // input character
   int i=0;              // array index

   inp = getchar();      // get first character
   while (inp != '\n')   // loop until ENTER
   {
      inputstr[i] = inp;    // store character
      i++;                  // advance array index
      inp = getchar();      // get next character
   }
   inputstr[i] = '\0';   // place end-of-string null char; !!! important !!!
}
```