CSE220
Spring 2014 – Practice Midterm #1

1. (Show all work) Convert the following binary number 010011101011 to:
   (a) Decimal

   $010011101011 = 2^{10} + 2^7 + 2^6 + 2^5 + 2^3 + 2^1 + 2^0 = 1024 + 128 + 64 + 32 + 8 + 2 + 1 = 1259_{10}$

   (b) Octal (base 8)

   Group by 3's ($2^3 = 8$) from right 010 011 101 011 2353_8

   (c) Hexadecimal (base 16)

   Group by 4's ($2^4 = 16$) from right 0100 1110 1011 $4EB_8$

   (d) Base 5

   Convert to decimal 1st. Then divide by 5.
   1259/5 = 251 R4
   251/5 = 50 R1
   50/5 = 10 R0
   10/5 = 2 R0
   2/5 = 0 R2 $= 20014_5$

   To check answer: $2*5^4 + 1*5^1 + 4*5^0 = 2*625 + 5 + 4 = 1259$

2. Convert 4.9 in decimal to a number in base 3. Stop after pattern starts repeating.

First convert integer part. Decimal 4 is 11 in base 3. To convert fractional part, multiply it by 3 and collect overflow.

0.9 0.22002200 Answer is 11.22002200...
x 3

2.7 --> Collect 2
0.7
x 3

2.1 --> Collect 2
0.1
x 3

0.3 --> Collect 0

0.3
x 3

0.9 --> Collect 0, Since we get 9 again, after this pattern will repeat.

3. Convert the decimal number $(0.563)_{10}$ to a binary number. Stop after 5 decimal places.

Multiply the number by the base of the new number reserving the value in the 1's position.
*0.563*2 = 1.126 -> 1*
*0.126*2 = 0.252 -> 0*
*0.252*2 = 0.504 -> 0*
*0.504*2 = 1.008 -> 1*
*0.008*2 = 0.016 -> 0*
*0.016*2 = 0.032 -> 0*
*0.032*2 = 0.064 -> 0*
….. *Continue until get 0, or until the pattern repeats.*

0.1001000…..

In this case, there is no stopping point. ON the exam, it will tell you how many positions to calculate to.

4. Write -12_{10} as a 6-bit (a) sign magnitude, (b)1's complement number, (c) 2's complement number.

(a) First, write 12_{10} in binary. 001100,
change most significant bit to 1 for negative number 101100

(b) First, write 12_{10} in binary. 001100
Then, flip bits 110011

(c) First, write 12_{10} in binary. 001100
Then, flip bits 110011
Add 1 110100

5. Normalize this binary number (-0.0001101). You must follow normalization of single precision IEEE format. State the sign bit, exponent, and significand in binary form for the above normalized number.

-1.101 x 2^-4
Sign bit is 1, exponent is 127-4=123, which is 01111011 in binary. Significand is 101000....00.

6. Consider 6-bit long (including sign) binary numbers.
   (a) Consider a binary number (110010) stored in signed magnitude form. What is this number in decimal?
   (b) How is -5 stored in 1's complement form?
   (c) Perform (001001 + 001101). Answer:
   (d) What is the smallest positive number that we need to subtract to (111010) so that negative overflow is generated? Decimal answer is expected; remember numbers are 6-bit long. Here negative numbers are stored in 2's complement form. Show all work.
   (e) Sign extend 110010 to an 8-bit number

   (a) -18,
   (b) 111010,
   (c) 010110
   (d) These are 6-bit long numbers including sign. So range is -32 to +31 (2^5 = 32). We have a negative overflow, if the result is -33 or beyond. The smallest number is (-33+6 = -27), and if we subtract 27, we will have -33 as an answer, a negative overflow.
   (e) Sign extended 8-bit number is 11110010.

7. (a) Suppose we want to code letters A through Z (only upper case) using binary numbers, starting with 0. What is the minimum number of bits that we need?
   (b) Suppose we want to code letters A through Z (only upper case) using base 3 numbers, starting with 0. What is the minimum number of base 3 digits that we need?
   (c) What would be the base-3 code for letter F in the part (b) above?

   (a) There are 26 options, therefore we need 5 bits to code A through Z.
   (b) There are 26 options, therefore we need 3 base-3 digits to represent 26 values uniquely
   (c) F is represented by value 5 (A is 0, B is 1, C is 2, etc). To encode 5 in Base 3= 012_3

8. Consider a machine with 32-bit word (4 bytes in a word). Bytes are numbered 0, 1, 2, 3, ... and words are numbered 0, 1, 2, ... etc. Word-0 contains byte-0, byte-1, byte-2 and byte-3. Word-1 contains byte-4, byte-5, byte-6, and byte-7 etc.
   (a) A byte numbered 406 would belong to what word? _____ (word-number)
   (b) Assume, this memory is organized using big endian byte order, what would be the least significant byte (one that holds the Least Significant Bit) of Word-20? _____ (byte-number)

   (a) A byte numbered 406 would be in a word numbered (406)/4 = 101 (integer division).
   *(b) The number of the LSByte of word 20 is byte-83. (20*4 + 3)*

9. Consider the following main function of a C program. Assume that memory is byte addressable, with big-endian byte order. Variables are allocated on stack, and stack grows from high address 250100 (in decimal) to some low numbered address.

```
int main ( ){
    char c1;              /* a character */
    double z;             /* a double */
    char c2;              /* another character */
    int x1, x2, x3;       /* three integers.  */
    int *px; double *pz;

    px = &x2;  pz = &z;   <============== (A)
    px++;  pz++;          <============== (B)

 }
```

(a) What would be the contents of px and pz after the instructions on line (A) are executed?
(b) What would be the contents of px and pz after the instructions on line (B) are executed?

Memory will be laid out as follows

Address of byte0	*Contents of byte0 byte1 byte2 byte3*
250100	*c1 -- -- --*
250096	*<---------- z ---------->*
250092	*<---------- z ---------->*
250088	*c2 -- -- --*
250084	*<---------- x1 ---------->*
250080	*<---------- x2 ---------->*
250076	*<---------- x3 ---------->*
250072	*<---------- px ---------->*
250068	*<---------- pz ---------->*

(a) px = 250080 pz = 250092
(b) px = 250084 pz = 250100

10. Consider the following C program. Here main calls function mystery twice.

```
#include <stdio.h>

void mystery(int *p1, int *p2){
  int x = 10, y = 5;
  *p1 = x + *p2;
  *p2 = *p1 + y;
  }
```

```
int main(void){
   int x = 2, y = 1;
   mystery(&x, &y);
   mystery(&y, &x);
   printf("x = %d    y = %d\n", x, y);
   return 0;
 }
```

What values are printed for x and y at the end?

x = 26 y = 21

11. Consider an array A of 10 integers. Assume we want to swap the contents of A[1] and A[2], then A[2] and A[3], then A[3] and A[4], then A[4] and A[5], …., A[8] and A[9]. Write C code to do the swaps using a loop and ONE integer pointer, ptr. DO NOT use any A[] references or any additional variables.

One approach is

```
int *ptr;

for(ptr=A+1; ptr <= A+8;ptr++)
{
        int temp = *ptr;
        *ptr = *(ptr +1);
        *(ptr+1) = temp;
}
```

12. Consider the following C program. It processes characters of `in' string one by one. If a character is an upper case letter, it writes four different characters to `out' string. If it is not, it writes the same in-string character 4 times to output string. This program has one while-loop with one if-then-else statement inside.

```
int main ( ) {

  char *in = "1M@Y";    /* Declare in string. */
  char c, out [100];    /* Out string of size 100 */
  int i, j;

  c = *in;      /* first character of `in' string. */
  i = 0;        /* Index for `out' string */

  while (c != '\0') {

    if (c >= 'A' || c <= 'Z')  <---  /* Test for upper case */
```

```
            for (j = 0; j < 4; j++) { /* Write 4 characters to
                                                 out string */
              out [i+j] = c+j;
              /* Check the written character.  */
              if (out[i+j] > 'Z') out[i+j] = out[i+j] - 'Z' + 'A'-1;
            }
       else
            for (j = 0; j < 4; j++)  /* Write the same character
                                                   4 times. */
               out [i+j] = c;        /* End of if-then-else */

     i= i+4; in++;  /* Increment pointers */
     c=*in;           /* Next character of in string */

   } /* end while */

   out[i] = '\0';  /* Terminate out string */
   printf ("%s\n", out);
   return 0;
}
```

(a) Is there a logical error in the condition of if statement indicated by left-arrow? Answer: Y N
(b) If yes, fix the error. If no, explain why?
(c) What string is printed at the end (with logical error corrected if needed)?

(a) Yes
(b) Condition should be (c >= 'A' && c <= 'Z')
(c) For upper case letters, it adds j = 0, 1, 2, and 3 and stores these 4 characters to out string. If it exceeds Z, it raps around and starts with A again. Answer: 1111MNOP@@@@YZAB (after it has been fixed.)