UMass Boston CS 240 – Spring 2020
Test 3
Time: 11:30 - 2:30, 05/19/2020

Submission

- Way of submission:
  1. The full points are 100, the extra 10 are bonus points.
  2. You can submit your test3 source code files into the exact folder `"cs240/test3"` within your home directory.
  3. Also you can submit your **`test3.c`** file named `[fullname_UnixAcountName]` to the email address of `cs240.umb@gmail.com`.
  4. The subject of the email should be `test3_spring2020`.
- Late submission: No late submission for test3.

Requirements

- Finish the code stubs as required.
- Write your code with necessary comments and proper spacing, especially when you don't think you will get the full score.
- Partial scores will be given when you clearly express what you have done for the different snippets of your code.
- If your program does not compile or run and your code is hard to read, then a lower score is guaranteed.

1 (20 points) – 2's complement format and Macros

- 1.1 (10 Points) Compile the following program into q1.out and run with the command **./q1.out q1.out**. Answer the following question:

```
/*
 * project: test3
 * file: q1.c
 */
#include <stdio.h>
int main(int argc, char *argv[]){
   printf("%d", argc);
   return 0;
}
```

  1. What's the output of this program (ie, the value of **argc**) when run with **./q1.out q1.out**? (2 Points)
  2. What's the 2's complement format of **-argc**. (8 Points)

- 1.2 (10 Points) Writing macros with argument.

  1. Define a macro to get the minimum value of two arguments.

     ```
     #define MIN(X,Y) [expression]
     ```

  2. Define a macro to get the multiplication of two arguments.

     ```
     #define MUL(X,Y) [expression]
     ```

2 (20 points)–Complicated Declaration

- (10 points) Write out each of their Postfix format.

  ```
  2.1(5 Points) char (*(*(*ok)(int))[5])();

  2.2(5 Points) float *(*(*right)())(int);
  ```

- 2.3 (10 Points) Initialize the **a1, a2** using values related to **a** in this function below.

  (Does not matter which values related to **a** specifically, but the syntax has to be right.) For example, int b = a[0][2] or int b1 = a[1][2] are all correct initialization.

  ```
  Another example:  int c=3; int *d=&c; this is correct initialization
  but int *d = c; is a wrong initialization.

  If your initialization pass the compiling all right, you will get all the points.
  If you just write down the correct datatype of a1 and a2 in comments you will get partial
  points.

  void q2(void){
      int a[2][3];
      // int *a1[3] = [?];
      // int (*a2)[3] = [?];
  }
  ```

3 (25 Points) Char array

1. (10 Points) Complete the function `char * capitalize(char *s)` that capitalize each word in the string **s** in place. For example, "hello world!" to "Hello World!". The words here are only made of both upper and lower case letters; all other characters are punctuation characters.

```
char * capitalize(char *s){


    return s;
}
```

2. (15 Points) Complete the function **Words split(const char *s, char c)** below that split the string **s** into a string array **words.words**. Specifically,

   - The **s** should be split into sub-strings which are separated by the character **c**.
   - All the sub-strings should be stored into the string array **words.words** which should be dynamically allocated memory just about the right proper size to store each sub-string.
   - For example, **split("hello world!", ' ')**, (the second argument is an empty space); the returned string array **words.words** should look like {"hello","world!"}, with all the memory allocated just enough to hold each sub-string.
   - The **words.num** should be the number of sub-strings.
   - The char array **s** is a constant char array, you should not change any character in it. It will cause error.

```
typedef struct
{
    char **words;
    int num;
} Words;


Words split(const char *s, char c){
    Words words;

    return words;
}
```

4 (20 points)

Define these functions below based on the structure of Matrix and Mat. Specifically,

- The average is the average value of the whole matrix, the row_sum the vector of sums of all row vectors, and the col_sum the vector of sums of all column vectors.

```
Example,   5, 7, 9  <- col_sum

           1, 2, 3      6
           4, 5, 6      15
                        row_sum
average = (1+2+3+4+5+6) / 6 = 3.5
```

- The pointer members need to be allocated memory properly for holding the their values. And here we use structure variables, not pointers to structure.
- For createMat function, allocate adequate memory for the members of pointers, and update their value properly based on the the argument Matrix mat passed in.

```
typedef struct
{
    int height, width;
    double **map;
} Matrix;


Matrix createMatrix(int height, int width); (6 Points)
void deleteMatrix(Matrix mat); (4 Points)



typedef struct
{
    Matrix *mat;
    double *row_sum, *col_sum, average;
} Mat;


Mat createMat(Matrix mat); (10 Points)
```

5 Comparator and Endianess (15 Points)

1. (5 Points)Write a comparator function to compare two Mat variables with their averages.

   ```
   int com_Mat(const void *a,  const void *b){

   }
   ```

2. (5 Points)Write a comparator function to compare two pointer variables to Mat variables with their averages.

   ```
   int com_PMat(const void *a,  const void *b){

   }
   ```

3. (5 Points).

   ```
   typedef union {
       unsigned int val;
       unsigned char c;
   } IP;

   void q5(void){
       IP ip;
       ip.val = 0x0A0B0C0D;
       // ip.c: ?
   }
   ```

   what is the integer value of **ip.c** above on a big-endianess machine ?

6 (10 points) LinkedList

```
typedef struct list_node
{
    int val;
    struct list_node *next;
}ListNode;

ListNode * swap(ListNode *head, int m, int n) {

    return head;
}
```

Complete the function **ListNode * swap(ListNode *head, int m, int n)** to swap the m-th and n-th node. (The index of LinkedList here starts from 0).

You should not swap the values in the list's nodes, only node references should be swapped.