Information and Computer Science Department
Second Semester 142
ICS 103 – Computer Programming in C

Final Examination
Thursday, May 21, 2015
Duration: 120 minutes

Name: KEY

ID#: | | | | | | | | | |

Please tick your section:

Instructor	Section
Dr. Muhammad Mudawar	[ ] 07(UT1100) [ ] 09(UT1310)
Dr. Samer Arafat	[ ] 17(MW0900) [ ] 20 (MW 1100) [ ] 23 (MW 1310)
Dr. Rafi Ul Hasan	[ ] 04 (UT 0800) [ ] 10 (UT 1310)
Dr. Basem Al-Madani	[ ] 01 (UT 0700) [ ] 05 (UT 0800) [ ] 08 (UT 1100)
Dr. Abdulaziz Alkhoraidly	[ ] 02 (UT 0700) [ ] 14 (MW0800)
Dr. Mohammed Balah	[ ] 03 (UT 0700) [ ] 06 (UT 0800) [ ] 11 (MW 0700) [ ] 16 (MW 0800) [ ] 19 (MW 0900)
Mr. Jaweed Yazdani	[ ] 21 (MW 1100) [ ] 24 (MW 1310)
Mr. Said A. Muhammad	[ ] 12 (MW 0700) [ ] 15 (MW 0800)
Mr. Hakim Adiche	[ ] 22 (MW 1100)
Mr. Hazem Selmi	[ ] 18 (MW 0900)

Instructions:
1. Answer all questions. Make sure your answers are clear and readable.
2. The exam is closed book and closed notes. No calculators or any helping aides are allowed. Make sure to turn off your mobile phone and keep it in your pocket.
3. If there is no space on the front of a question's page, use the back of the page. Indicate this clearly.

Question #	Maximum Grade	Obtained Grade	Remarks
1	20		
2	20		
3	10		
4	15		
5	15		
6	20		
Total	100		

Question # 1 [20 points] Choose the **most correct** answer for each of the following:

1. If infile is a FILE pointer to an input data file, to check whether the file has **NOT** been opened successfully which of the following statements is the most correct?

   A. if(infile == NULL) printf("Sorry, input file not found"); A
   B. if(infile == -1) printf("Sorry, input file not found");
   C. if(infile == FileOpenError) printf("Sorry, input file not found");
   D. if(infile == EOF) printf("Sorry, input file not found");
   E. All of the above are correct.

2. Given the following function definition header:

   void myfunction(int a, int b, int *prod)

   and given the following related function call:

   myfunction(a, b, &prod) ;

   where **a**, **b**, and **prod** are integer variables. Which one of the function definition bodies, below, is the one that would properly compute and return the product of inputs **a** and **b ?** Assume that **a** and **b** are initialized.

   A. {int prod; prod = a * b; }
   B. {int prod; prod = a * b; return prod; }
   C. {*prod = a * b ; } C
   D. {return a*b ; }
   E. None of the above is correct.

3. Given the program below, we would like to insert a new statement, after the statement **p = &array1[2][2]**, that would assign the integer value 5 to the 2-D array element indexed by row # 2 and column # 2:

```
#include<stdio.h>
#define ROWS 3
#define COLS  3
int main(void) {
    int array1[ROWS][COLS] ;
    int *p ;
    p = &array1[2][2] ;
    ---------------------------- // new statement to be inserted here
    printf("%d", array1[2][2]) ;
   return 0;
 }
```

   Which of the following answers is the best?

   A. array1[2][2] = 5 ;
   B. *p = 5 ;
   C. array1[ROWS-1][COLS-1] = 5 ;
   D. All of the above are correct. D
   E. None of A, B, and C are correct.

4. An integer 2-D array, **arr**, has 4 rows and 3 columns. Which of the following will correctly initialize all **arr** elements to 1?

A. `int arr[4][3] = {1} ;` **C**
B. `int arr[4][3] = { {1} , {1} , {1} , {1} } ;`
C. `int arr[4][3], i, j; for(i=0; i<4; i++) for(j=0; j<3; j++) arr[i][j] = 1;`
D. `int arr[4][3], i, j; for(j=0; j<3; i++) for(i=0; i<4; j++) arr[j][i] = 1;`
E. None of the above is correct.

5. What is the output of the following program?

```
#include<stdio.h>
#include<string.h>
int main(void) {
   int k ;
    char string1[] = { 'I', 'C', 'S', ' ', '1', '0', '3', '\0' } ;
    k =  strlen(string1) ;
    switch(k){
      case 6: printf("A") ;
              break;
      case 7: printf("B") ;
              break;
       case 8: printf("C") ;
              break;
      default: printf("D") ;
    }
    return 0;
}
```

A. A
B. B **B**
C. C
D. D
E. None of the above is correct.

6. The function call **`tolower('n')`**:

A. Generates an error since `'n'` is already in lower case.
B. Returns the character `'m'`
C. Returns the character `'N'`
D. Returns the character 'n' **D**
E. Returns the null character `'\0'`

7. Consider the following array: `int x[]={2, 4, 6, 7, 9, 13};`
Using the linear search function, how many comparisons will be conducted if the target value is 24?

A. 0
B. 1
C. 5
D. 7
E. 6 **E**

8. A 2-D array **A** has **m** rows and **n** columns, and a 2D-array **B** has **x** rows and **y** columns. To get the absolute value of the difference (subtraction) of all the elements of **A** minus **B**, or, mathematically, to compute **|A – B|** which one of the following statements must be true regarding the number of rows and columns of the two arrays?

A. x == m and y == m
B. x == m and y == n B
C. x == n and y == n
D. x == n and y == m
E. None of the above is correct.

9. What is the output of the following program?

```
#include<stdio.h>
#include<string.h>

int main(void) {
   char string1[] = "CE 324";
   char string2[] = "ME 101";
   if(strcmp(string1,string2) == 0)
      printf("A") ;
   else if (strcmp(string1,string2) > 0)
      printf("B") ;
   else
      printf("C") ;

   return 0;
}
```

A. A
B. B
C. C C
D. The information given is not sufficient to give an answer.
E. None of A, B, and C is correct.

10. Given the following function prototype:

```
void getAverageAndSum(double a[], double b[],int size, double *p, double *e);
```

Which of the function calls below is correct for the above function prototype? Assume **x** and **y** are 1-D arrays of type double. Assume **SIZE** is integer, and **sum** and **average** are variables of type double.

A. getAverageAndSum(x, y, SIZE, &sum, &average); A
B. `getAverageAndSum( x[SIZE], y[SIZE], SIZE, &sum, &average);`
C. `getAverageAndSum( &x, &y, SIZE, &sum, &average);`
D. `getAverageAndSum( x, y, SIZE, sum, average) ;`
E. None of the above is correct.

Question # 2 [20 points]

Write the output of each of the following C programs or program fragments in the empty box below each program or program fragment:

Program fragment 1 (5 points):

```
int x[5], k;
for(k = 0; k <= 4; k++){
   if( k % 2 == 0)
       x[k] = k + 2;
   else{
       x[k – 1] = k + 3;
       x[k] = x[k – 1] + 4;
   }
 }

for(k = 4; k >= 0; k--)
     printf("%d   ",  x[k]);
```

Output:

6 10 6 8 4

Program fragment 2 (5 points):

```
int matrix[][4] = {{5, 12, 8, 25}, {9, 6, 4, 0}, {3, 7, 1, 10}}, k, m;
for(k = 2; k >= 1; k--){
    for(m = 3; m >= 0; m--){
         printf("%d  ", matrix[k][m]);
    }
    printf("\n");
 }
```

Output:

10 1 7 3
0 4 6 9

Program 3 (5 points):

```
#include <stdio.h>
#include <string.h>

int main(void){
      char str[81] = "HE LIKES ICS 103";
      char *token;
      int k;

      token = strtok(str, " ");
      int count = 1;
      do{
            printf("%d  ", count);
            for(k = strlen(token) - 1; k >= 0; k--){
                  printf("%c", token[k]);
            }

            printf("\n");
            count++;
            token = strtok(NULL, " ");
      }while(token != NULL);

      return 0;
}
```

Output:

```
1   EH
2   SEKIL
3   SCI
4   301
```

Program 4 (5 points):

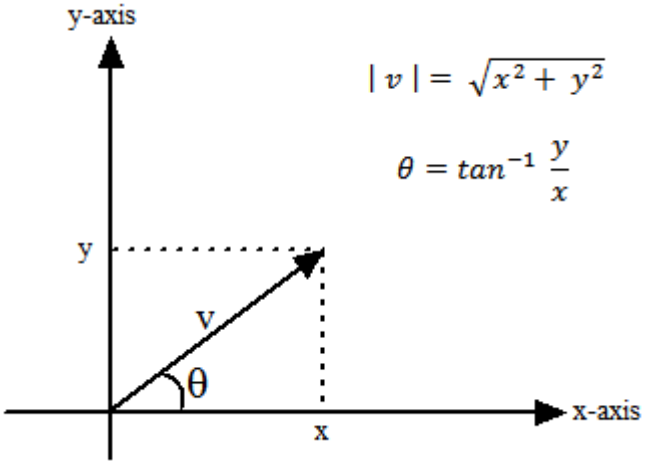
```
#include <stdio.h>
#include <string.h>

int main(void){
    char str[2][80] = {"ICS 103", "finalexam"};
    int k;
    for(k = 0; k < strlen(str[1]); k++){
       if(isalpha(str[0][k]))
         str[0][k] = toupper(str[1][k]);
       else if(isdigit(str[0][k]))
         str[0][k] = str[1][k];
       else if(isspace(str[0][k])){
         str[0][k] = str[1][1];
         str[1][k] = '$';
       }
     }

     for(k = 0; k <= 1; k++)
        puts(str[k]);

        return 0;
}
```

Output:

FINilex
fin$lexam

Question # 3 [10 points]

Write a C function that takes as input the x-component and the y-component of a vector **v** that lies on the first quarter of the x-y plane. The function returns the vector magnitude **|v|** and the angle **θ**, in DEGREES, the vector makes with the x-axis.



$$|v| = \sqrt{x^2 + y^2}$$

$$\theta = tan^{-1}\frac{y}{x}$$

Example: For the vector $\vec{V} = < 3, 4 >$, the magnitude can be calculated using |V|= $\sqrt{3^2 + 4^2} = 5$ and the angle θ with the x axis can be calculated using $\theta = \tan^{-1}\frac{4}{3}$ = 53.1 degrees.

Note: The C function for $\tan^{-1}\frac{y}{x}$ is **atan(y/x)**. It returns an angle in radians. The relation between **radians** and **degrees** is given by the formula:

$$radians = \frac{\pi}{180} \times degrees$$

Note:

- Assume **x** and **y** are both positive (the vector lies in the first quarter)
- Your function must be general and not specific to the above example.
- YOUR FUNCTION MUST NOT CONTAIN ANY **scanf** AND **printf** STATEMENTS.
- Use 3.14159 as the value of **π**
- Do NOT write the main function.

Key:

```
void magnitudeAndAngle(double x, double y, double* magnitude, double* angle){
     double radians, PI = 3.14159;
     *magnitude = sqrt(x * x + y * y);
     radians = atan(y/x);
    *angle = radians*180/ PI;
 }
```

Question # 4 [15 points]

Write a C function that takes the number of rows **n**, the number of columns **m**, a target integer value **x** and a matrix (of size *n* * *m*) as input arguments. The function returns a 2D-array with two rows and n * m columns that contains the row and column indexes of all elements of the matrix that are equal to the value **x**. It also returns a **count** of such elements.

Example: If the input matrix is:

5	2	9	4	2
7	6	2	0	8
1	0	8	2	4

and **x** is 2, then the locations where **x** is found in the matrix are (0,1), (0,4), (1,2) and (2, 3), therefore the returned array is:

0	0	1	2											
1	4	2	3											

and the returned **count** is 4.

Note:

- Your function must be general and not specific to the above example.
- YOUR FUNCTION MUST NOT CONTAIN ANY **scanf** AND **printf** STATEMENTS.
- Do NOT write the main function.

Key:

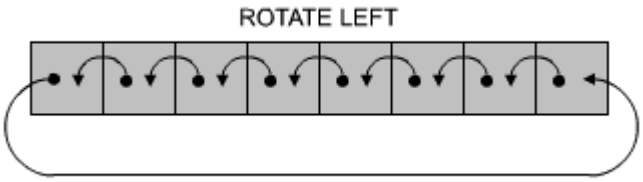
```
void locations(int n, int m, int x,int matrix[][m],int loc[2][n*m],int* count){

      int r, c, counter = 0;

      for(r = 0; r <= n - 1; r++){

         for(c = 0; c <= m - 1; c++){

             if(matrix[r][c] == x){

               loc[0][counter] = r;
               loc[1][counter] = c;

               counter++;
             }
          }
      }

      *count = counter;
}
```

Alternate Key:

```
void locations(int matrix[][COLS],int loc[2][ROWS*COLS],int n, int m, int x, int* count){
       int r, c, counter = 0;

       for(r = 0; r <= n - 1; r++){

          for(c = 0; c <= m - 1; c++){

              if(matrix[r][c] == x){

                loc[0][counter] = r;
                loc[1][counter] = c;

                counter++;
              }
           }
       }

       *count = counter;
}
```

Question # 5 [15 points]

Write a C function called ***rotateLeftN*** that takes a 1D-array of type double, the size of the array, and an integer value **N** as input arguments. The function modifies the passed array by rotating each element of the array to the left by an amount specified by **N**.



Example: If the passed array is:

1.0	5.0	3.5	8.0	12.0	4.6

then, a left rotation by an amount of **1** will result in:

5.0	3.5	8.0	12.0	4.6	1.0

And a left rotation by an amount of **2** will result in:

3.5	8.0	12.0	4.6	1.0	5.0

Note:

- Your function must be general and not specific to the above example.
- YOUR FUNCTION MUST NOT CONTAIN ANY **scanf** AND **printf** STATEMENTS.
- Assume that **N** is a positive value.
- Do NOT write the main function.

Key:

```
void rotateLeftN(double array[], int size, int N){

     double temp;
     int k, m;

     for(k = 1; k <= N; k++){

         temp = array[0];

         for(m = 0; m <= size - 2 ; m++){

            array[m] = array[m + 1];

          }

          array[size - 1] = temp;
     }
 }
```

Question # 6 [20 points]

Write a complete C program that reads a character from the keyboard. It then determines the frequency (the number of occurrences) of this character in each line of an input file **input.txt** together with the total frequency (the total number of occurrences) of the character. The output must be stored in a textfile **output.txt** in the format given in the sample program run below, where each output line contains a line number where the input character appears in addition to the frequency of that character in the line. The last output line contains the total frequency of the character. If the input character is not found, the output in the textfile will be the message: **The character is not found.**

Note:

- Your program must be general; it must work for any input character and any input file.
- Assume that the maximum number of characters in a line of the textfile is 80.

Sample **input.txt**

```
This is a test
Dhahran
is south of
Bahrain.
```

Sample program run:

```
Please enter a character:  s
```

Sample **output.txt**

```
Line#: 1,  Frequency of s = 3
Line#: 3,  Frequency of s = 2
Total frequency of s = 5
```

Key01: Single loop

```
#include <stdio.h>

int main(void){
        FILE *inptr, *outptr;
        char line[80], ch, inputchar;

        int lineNum = 0, totalFrequency = 0 , lineFrequency = 0, k;

        inptr = fopen("input.txt", "r");

        if(inptr == NULL){
                printf("Error in opening input file");
                exit(1);
        }

        outptr = fopen("output.txt", "w");

        printf("Enter a character: ");
        scanf("%c", &inputchar);

        while(fscanf(inptr, "%c", &ch) != EOF){
            if(ch != '\n'){
                if(ch == inputchar)
                   lineFrequency++;
            }
            else{
                lineNum++;

                if(lineFrequency != 0){
                   fprintf(outptr, "Line#: %d,  Frequency of %c = %d\n",
                       lineNum, inputchar, lineFrequency);
                   totalFrequency += lineFrequency;
                   lineFrequency = 0;
                }
            }
        }

        if(totalFrequency != 0)
          fprintf(outptr, "Total frequency of %c = %d", inputchar, totalFrequency);
        else
          fprintf(outptr, "The character %c is not present", inputchar);

        fclose(inptr);
        fclose(outptr);

        return 0;
}
```

Key02: Nested loops – With inner for-loop

```
#include <stdio.h>
#include <string.h>

int main(void){

        FILE *inptr, *outptr;
        char line[80], ch;

        int lineNum = 0, totalFrequency = 0 , lineFrequency, k;

        inptr = fopen("input.txt", "r");

        if(inptr == NULL){
                printf("Error in opening input file");
                exit(1);
        }

        outptr = fopen("output.txt", "w");

        printf("Enter a character: ");
        scanf("%c", &ch);

        while(fgets(line, 80, inptr) != NULL){
                lineNum++;
                lineFrequency = 0;
                for(k = 0; k < strlen(line); k++){
                   if(ch == line[k]){
                           lineFrequency++;
                   }
            }

            if(lineFrequency != 0){
                fprintf(outptr, "Line#: %d,  Frequency of %c = %d\n", lineNum, ch, lineFrequency);
                totalFrequency += lineFrequency;
            }
        }

        if(totalFrequency != 0)
          fprintf(outptr, "Total frequency of %c = %d", ch, totalFrequency);
        else
          fprintf(outptr, "The character is not present");

        fclose(inptr);
        fclose(outptr);

        return 0;
}
```

Key03: Nested loops – With inner while-loop

```
#include <stdio.h>

int main(void){
        FILE *inptr, *outptr;
        char line[80], ch;

        int lineNum = 0, totalFrequency = 0 , lineFrequency, k;

        inptr = fopen("input.txt", "r");

        if(inptr == NULL){
                printf("Error in opening input file");
                exit(1);
        }

        outptr = fopen("output.txt", "w");

        printf("Enter a character: ");
        scanf("%c", &ch);

        while(fgets(line, 80, inptr) != NULL){
                lineNum++;
                lineFrequency = 0;
                k = 0;
                while(line[k] != '\0'){
                   if(ch == line[k]){
                           lineFrequency++;
                   }

                   k++;
               }

               if(lineFrequency != 0){
                   fprintf(outptr, "Line#: %d,  Frequency of %c = %d\n", lineNum, ch, lineFrequency);
                   totalFrequency += lineFrequency;
               }
        }

        if(totalFrequency != 0)
          fprintf(outptr, "Total frequency of %c = %d", ch, totalFrequency);
        else
          fprintf(outptr, "The character %c is not present", ch);

        fclose(inptr);
        fclose(outptr);

        return 0;
}
```