

Information and Computer Science Department
 First Semester 161
 ICS 103 – Computer Programming in C
Major 02 Examination
 Saturday, December 24, 2016 (9:00-11:00AM)
 Duration: 120 minutes

Name:

KEY

 Lect Serial #

ID#:

--	--	--	--	--	--	--	--	--

Please tick your section:

Instructor	Section	
Mr. Said Abdallah Muhammad	<input type="checkbox"/> 01 (UT 7:00 - 7:50)	<input type="checkbox"/> 02 (UT 7:00 - 7:50)
Dr. Hamood Al-Jamaan	<input type="checkbox"/> 03 (UT 8:00 - 8:50)	<input type="checkbox"/> 04 (UT 8:00 - 8:50)
Dr. Rafiul Hassan	<input type="checkbox"/> 05 (UT 11:00 - 11:50)	<input type="checkbox"/> 06 (UT 11:00 - 11:50)
	<input type="checkbox"/> 07 (UT 13:10 - 14:00)	<input type="checkbox"/> 08 (UT 13:10 - 14:00)
Dr. Muhammad Balah	<input type="checkbox"/> 09 (MW 7:00 - 7:50)	<input type="checkbox"/> 10 (MW 7:00 - 7:50)
Mr. Muhammad Aslam	<input type="checkbox"/> 11 (MW 8:00 - 8:50)	<input type="checkbox"/> 12 (MW 8:00 - 8:50)
Dr. Samer Arafat	<input type="checkbox"/> 13 (MW 9:00 - 9:50)	<input type="checkbox"/> 14 (MW 9:00 - 9:50)
	<input type="checkbox"/> 15 (MW 10:00 - 10:50)	<input type="checkbox"/> 16 (MW 10:00 - 10:50)
Dr. Louai Al-Awami	<input type="checkbox"/> 17 (MW 11:00 - 11:50)	<input type="checkbox"/> 18 (MW 11:00 - 11:50)
	<input type="checkbox"/> 19 (MW 13:10 - 14:00)	<input type="checkbox"/> 20 (MW 13:10 - 14:00)

Instructions:

1. Answer all questions. Make sure your answers are clear and readable.
2. Make sure there are 5 questions in 10 pages.
3. 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.
4. If there is no space on the front of a question's page, use the back of the page. Indicate this clearly.

Question #	Max Grade	Obtained Grade	Remarks
1	15		
2	25		
3	15		
4	15		
5	30		
Total	100		

Question 1 [15 points]

Part 1 (10 points):

Select the correct answer by filling the table below:

1	2	3	4	5
C	C	A	C	D

1. A pointer is
 - A. A keyword used to create variables
 - B. A variable that stores the address of an instruction
 - C. A variable that stores the address of another variable
 - D. All of the above.
2. Which of the statements is correct about the following C program?

```
#include<stdio.h>
int main()
{
    int i = 100;
    int *j = &i;
    return 0;
}
```

- A. **j** and **i** are pointers to an **int**
 - B. **i** is a pointer to an **int** and stores the address of **j**
 - C. **j** is a pointer to an **int** and stores the address of **i**
 - D. **j** is a pointer to a pointer to an int and stores the address of **i**
3. An array elements are always stored in _____ memory locations.
 - A. Sequential
 - B. Random
 - C. Sequential and Random
 - D. None of the above

4. What will happen if in a C program if you assign a value to an array element whose subscript (or index) exceeds the size of the array?
- The element will be set to 0.
 - The compiler would report an error.
 - The program may crash if some important data gets overwritten.
 - The array size would appropriately grow.
5. The prototype of a function that takes two arrays of type **double** as input arguments and returns an array of type **double** (All of size **s** which is a variable) can be written as:
- `double fun(double a[],double b[],double *c[], int s);`
 - `void fun (double a[s],double b[s],double *c[s]);`
 - `void fun (double a[s],double b[s],double c[s]);`
 - `void fun(double a[],double b[],double c[], int s);`

Part 2 (5 points):

Using the selection sort algorithm, show the content of the array below at the end of each of the first two passes done by the algorithm to sort the following array in an increasing order (i.e. from lowest to highest):

23	Pass1	Pass2
4	2	2
70	4	4
2	70	70
12	23	23
	12	12

Question 2 [25 points]

What will be the output of the following C code fragments or programs?

Code	Output
<pre> int num[5] = {1, 2, 20, 2, 30}; int x, y, z; x = ++num[1]; y = num[1]; z = num[++x]; printf("%d, %d, %d\n", x, y, z); </pre> <p>(6 points)</p>	<p>4, 3, 30</p>
<pre> int k=5, *j; j = &k; printf("%d\n", k**j+*j); </pre> <p>(3 points)</p>	<p>30</p>
<pre> #include <stdio.h> int fun(int *j, int k); int main() { int k=3, j=5; j = fun(&k, j); printf("%d %d", k,j); return 0; } int fun(int *j, int k) { *j = *j*5; return k*k; } </pre> <p>(4 points)</p>	<p>15 25</p>
<pre> int k,j; k=5; while(k<=9) { for (j=k;j>=7;j=j-3) printf("- "); printf("+\n"); k=k+2; } printf("%d\n",k); </pre> <p>(7 points)</p>	<p>+ - + - + 11</p>
<pre> int num[5], k; for(k = 0; k <= 4; k++){ if(k % 2 == 0) num[k] = k + 2; else{ num[k - 1] = k + 3; num[k] = num[k - 1] + 4; } } for(k = 4; k >= 0; k--) printf("%d ", num[k]); </pre> <p>(5 points)</p>	<p>6 10 6 8 4</p>

Question 3 [15 points]

A text-file **data.txt** contains the scores of students in an exam; each score is of type **double**. Write a C program that reads the scores, and then it computes and writes on the screen the percentage of the scores above 75.

Note:

- Your program must handle file not found case properly.
- Your program must be general; it must work for any number of scores in **data.txt**

Sample **data.txt**

```
50.0 76.0
40.0
80.0 95.0 35.0
60.0
50.0
66.0
45.0
```

Sample output:

30.00 percent of the scores are above 75.0

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    double score,perc;
    int tot, count, status;
    count = 0;
    tot = 0;
    FILE *infile;
    infile = fopen("data.txt","r");

    if(infile == NULL) {
        printf("File not found\n");
        exit(1);
    }

    status = fscanf(infile,"%lf",&score);
    while(status!= EOF) {
        tot = tot + 1;
        if(score > 75)
            count = count + 1;
        status = fscanf(infile,"%lf",&score);
    }
    perc = (double) count / tot * 100;
    printf("%.2f percent of scores are above 75\n",perc);

    fclose(infile);
    return 0;
}
```

Question 4 [15 points]

Write a function **findNumbers** that receives two values of type **double**: the **product** of two numbers x and y (product = $x * y$) and the **quotient** of those two numbers (quotient = x / y), it then calculate and return the two numbers x and y , each of type **double**. Write also the **main** function that prompts for and reads **product** and **quotient** and passes them to the **findNumbers** function. Your **main** function finally displays the two values returned by **findNumbers**.

Note:

- Use valid product, quotient pairs for your input [Don't test this].
- Assume that none of the input is zero or negative [Don't test this].
- The **findNumbers** function must be written after the main function.
- Don't use variables declared outside functions.

Sample program run:

```
Enter the product and quotient of two numbers: 10.0 2.5
num1 = 5.000000, num2 = 2.000000
```

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

```
void findNumbers(double product, double quotient, double* num1, double* num2);
```

```
int main(void){
    double product, quotient, num1, num2;

    printf("Enter the product and quotient of two numbers: ");
    scanf("%lf%lf", &product, &quotient);
    findNumbers(product, quotient, &num1, &num2);
    printf("\nnum1 = %f, num2 = %f\n", num1, num2);
    return 0;
}
```

```
void findNumbers(double product, double quotient, double* num1, double* num2){

    *num2 = sqrt(product / quotient);
    *num1 = product / *num2;
    return; // optional
}
```

Question 5 [30 points]

Definition:

Given two 1-D arrays **a** and **b** that have the same size:

a = [**a**₀, **a**₁, **a**₂, . . . , **a**_{N-1}] and

b = [**b**₀, **b**₁, **b**₂, . . . , **b**_{N-1}]

the dot product is defined as:

$$\sum_{i=0}^{N-1} \mathbf{a}_i * \mathbf{b}_i = (\mathbf{a}_0 * \mathbf{b}_0) + (\mathbf{a}_1 * \mathbf{b}_1) + (\mathbf{a}_2 * \mathbf{b}_2) + \dots + (\mathbf{a}_{N-1} * \mathbf{b}_{N-1})$$

Example:

The dot product of [1 2 3] and [4 5 6] arrays = 1*4 + 2*5 + 3*6 = 32.

Problem Statement:

Given a 1-D array **S** of size 1000, and another 1-D array **W** of size 10, a third 1-D array, **T** of size 991, is created as follows:

A dot product is computed using **W** and the first 10 elements of **S**, i.e., S[0], S[1], S[2], ..., S[9] and the result of the dot product is stored in the array element **T[0]**.

Next, a dot product is computed using **W** and the next ten **S** elements, that is, with the indices of **S** starting at 1, i.e., S[1], S[2], S[3], ..., S[10]. The result of the dot product is stored in **T[1]**.

Next, a dot product is computed using **W** and the next ten **S** elements, that is, with the indices of **S** starting at 2, i.e., S[2], S[3], S[4], ..., S[11]. The result of the dot product is stored in **T[2]**.

This process continue until **T[990]** is formed from the dot product of **W** and S[990], S[991], S[992], ..., S[999].

Write a C program to create the array **T**. Your program must have the following two functions:

1. **dotProduct** that computes the dot product of two input 1-D arrays.
2. **transform** that creates array **T** given input arrays **S** and **W**.

Include in your program the statement to initialize **W** with the following 10 values:

1.0, 2.0, 3.0, -2.0, 9.0, -2.0, 3.0, 2.0, 1.0, 0.0

Initialize the array **S** by reading 1000 values of type **double** from an input file **signal.txt**. The contents of the output array **T** must be stored in an output file **transform.txt**.

Note:

- Your program must be general.
- Your program must not use variables declared outside functions.
- The functions **dotProduct** and **transform** must not contain input / output function calls, like scanf, fscanf, printf, fprintf.
- The function **dotProduct** and **transform** must be written after the **main** function.

Solution 01:

```

#include <stdio.h>
#define SIZE_W 10
#define SIZE_S 1000
#define SIZE_T 991
double dotProduct(double array1[ ], double array2[ ], int size);
void transform(double S[ ], double W[ ], double T[ ]);
int main(){
    double W[ ] = {1.0, 2.0, 3.0, -2.0, 9.0, -2.0, 3.0, 2.0, 1.0, 0.0};
    double S[SIZE_S], T[SIZE_T];
    int k, m;
    FILE *SFileptr, *TFileptr;
    if( (SFileptr = fopen("signal.txt", "r")) == NULL){
        printf("Error in opening wavelet.txt");
        return 1;
    }
    TFileptr = fopen("transform.txt", "w");
    m = 0;
    while(fscanf(SFileptr, "%lf", &signal[m]) != EOF)
        m++;
    transform(S, W, T);
    for(k = 0; k < SIZE_T; k++)
        fprintf(TFileptr, "%0.2f ", T[k]);
    fclose(SFileptr);
    fclose(TFileptr);
    return 0;
}

double dotProduct(double array1[], double array2[], int size){
    double sum = 0;
    int k;
    for(k = 0; k <= size - 1; k++)
        sum += array1[k] * array2[k];
    return sum;
}

void transform(double S[ ], double W[ ], double T[ ]){
    double subArray[SIZE_W];
    int i, n ;
    for(i = 0; i <= SIZE_T - 1; i++){
        for(n = 0; n <= SIZE_W - 1; n++){
            subArray[n] = S[n + i];
        }
        T[i] = dotProduct(subArray, W, SIZE_W);
    }
}

```

Solution 02:

```

#include <stdio.h>
#define SIZE_W 10
#define SIZE_S 1000
#define SIZE_T 991
double dotProduct(double array1[ ], double array2[ ], int size);
void transform(double S[ ], double W[ ], double T[ ] );
int main(){
    double W[ ] = {1.0, 2.0, 3.0, -2.0, 9.0, -2.0, 3.0, 2.0, 1.0, 0.0};
    double S[SIZE_S], T[SIZE_T];
    int k, m;
    FILE *SFileptr, *TFileptr;
    if( (SFileptr = fopen("signal.txt", "r")) == NULL){
        printf("Error in opening wavelet.txt");
        return 1;
    }
    TFileptr = fopen("transform.txt", "w");

    m = 0;
    while(fscanf(SFileptr, "%lf", &S[m]) != EOF)
        m++;
    transform(S, W, T);
    for(k = 0; k < SIZE_T; k++)
        fprintf(TFileptr, "%0.2f ", T[k]);
    fclose(SFileptr);
    fclose(TFileptr);
    return 0;
}

double dotProduct(double S[], double W[],int start){
    int i;
    double sum = 0;
    for(i = 0; i < SIZE_W; i++)
        sum = sum + S[start + i] * W[i];
    return sum;
}

void transform(double S[ ], double W[ ], double T[ ] ){
    int i, n ;
    for(i = 0; i <= SIZE_T - 1; i++){
        T[i] = dotProduct(S, W, i);
    }
}

```