King Fahd University of Petroleum and Minerals
Information and Computer Science Department
ICS 103: Computer Programming in C
Summer Semester 2008-2009 (Term 083)
Major Exam II

Time: 120 minutes **Wednesday August 19, 2009, 7:30 PM**

Name:	KEY
ID#:	

PLEASE CIRCLE YOUR SECTION BELOW:

Section	1	2	3
Time	**SUMT 8:10-9**	**SUMT 9:20-10:10**	**SUMT 9:20-10:10**

Note:

- **Copying or Discussion will result in zero grade for all the students involved.**
- **Attempt all questions.**

Question #	Maximum Marks	Obtained Marks
1	12	
2	30	
3	20	
4	20	
5	18	
Total	100	

Good Luck

Question 1: (12 points)

Answer the following questions:

(1) The value printed by the following code fragment is 14.

```
int i=5, j=10;
printf("%d",i++ + --j);
```

(2) The value printed by the following program is 5.

```
#include <stdio.h>
void sqr (int i){ i=i*i;}
int main (void){
      int i=5;
      sqr(i);
      printf("%d",i);
      return 0;
}
```

(3) Determine the error in the following program:

```
#include <stdio.h>
int main (void){
      int i, *j;
      i=5; *j=i+1;
      printf("%d %d",i,*j);
      return 0;
}
```

The error is that j is not assigned any address and assigning a value to the content of j will lead to an error during the execution of the program.

(4) Determine the error in the following program:

```
#include <stdio.h>
#define SIZE 10
int main (void){
      int i, A[]={0,1,2,3};
      for (i=0; i<SIZE; i++)
            printf("%d",A[i]);
      return 0;
}
```

The error is that the array A is defined with 4 elements while when it is printed elements not defined in the array will be accessed i.e. A[4] to A[9]. This will lead to accessing data that was not allocated for the array.

Question 2: (30 points)

Find the output of each of the following programs:

Program	**Output**

(i) 6 points
 	
`#include<stdio.h>`
`void hill(int n){`
`  printf("%d ", n);`
`  if(n<=100){`
`  hill(3*n-1);`
`  printf("%d ", n);`
`  }`
`}`
 	
`int main(void){`
`  hill(1);`
`  return 0;`
`}`
	1 2 5 14 41 122 41 14 5 2 1

(ii) 6 points
 	
`#include <stdio.h>`
 	
`int test(int m, int n)`
`{`
 	
`  if (m == 0 && n == 0) return(-1);`
`  else if (m==0) return(n);`
`  else if (n==0) return(m);`
 	
`  else return(test(n, m%n));`
`}`
 	
`int main(void)`
`{`
`  int x=70, y=49;`
 	
`  printf("%d\n",test(x,y));`
`  return 0;`
`}`
	7

(iii) **6 points**

```
#include <stdio.h>

int main(void)
{
  int x=654, y;
  while (x>9){
    y=0;
    while (x>0){
      y += x%10;
      x /= 10;
    }
    x=y;
  }

  printf("%d\n",x);
  return 0;
}
```

6

(iv) **6 points**

```
#include<stdio.h>

int main(void){
  int n[8]={1, 1};
  int i;
  for(i=2; i<8; i++)
     n[i]=n[i-1]+n[i-2];
  for(i=0; i<8;i++)
     printf(“%d\n”, n[i]);

  return 0;
}
```

1
1
2
3
5
8
13
21

(v) **6 points**

```
#include <stdio.h>

int main(void)
{

  int n=3;
  for (int i=1; i<=n; i++){
      for (int j=1;j<=n-i; j++)
          printf(" ");
      for (int j=1;j<=2*i-1; j++)
          printf("*");
      printf("\n");
  }

  for (int i=n; i>=1; i--){
      for (int j=1;j<=n-i; j++)
          printf(" ");
      for (int j=1;j<=2*i-1; j++)
          printf("*");
      printf("\n");
  }

return 0;
}
```

```
  *
 ***
*****
*****
 ***
  *
```

Question 3: (20 points = 10 pts for (i) and 10 pts for (ii))

(i) The Ackermann function is defined recursively for non-negative integers ***m*** and ***n*** as follows:

$$A(m,n) = \begin{cases} n+1 & if\ m = 0 \\ A(m-1,1) & if\ m > 0\ and\ n = 0 \\ A\big(m-1, A(m,n-1)\big) & if\ m > 0\ and\ n > 0 \end{cases}$$

Write a function **ACKF** that receives two integer arguments ***m*** and ***n*** and returns the computed integer value ***A(m, n).***

```
int ACKF(int m, int n){
        if ( m == 0 ) return n+1;
        else if (m>0 && n==0) return ACKF(m-1,1);
        else return ACKF(m-1, ACKF(m, n-1));
}
```

(ii) The cosine function can be defined analytically using the infinite sum as follows:

$$\cos x = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n}}{(2n)!} = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \cdots$$

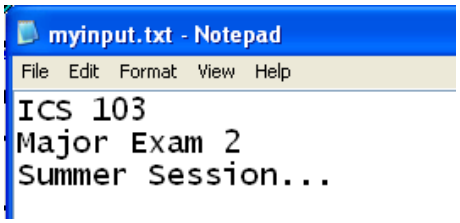
Write a function **cosine** that receives a real number ***x*** and returns a real number computing the cosine of ***x*** based on the above formula. The function also receives another integer ***m*** that determines the number of terms that will be used in computing the cosine function i.e. the higher the value of ***m*** the higher the accuracy.

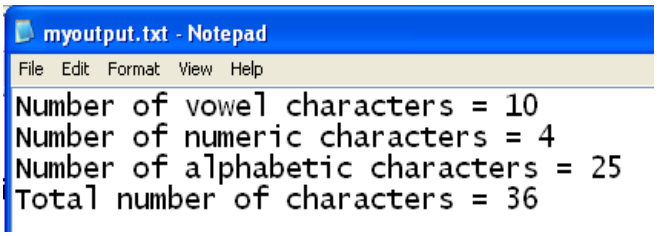
```
double cosine(double x,int m){

 int i;
double mcos=0;

for (i=0;i<m;i++)
        mcos += (pow(-1,i)*pow(x,2*i))/fact(2*i);
 return mcos;
}

double fact(double n) {
 if (n == 0)
        return 1;
else
        return  n*fact(n-1);
}
```

Question 4: (20 points)

Write a C program to open an input file named ***myinput.txt*** and count the number of vowel characters i.e. (a, e, i, o, u) both lower and upper case, the number of numeric characters (i.e. '0' to '9'), the number of alphabetic characters (i.e. 'a'...'z' and 'A'…'Z') and the total number of characters. The results should be printed in an output file named ***myoutput.txt***. Your program should handle file not found error. Minimize the number of statements used in your program. Do not use functions defined in ctype.h.

Sample input and output files:






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

int main(void)
{
        FILE *infile, *outfile;
        char c;
        int total=0, alpha=0, numeric=0, vowels=0;

        infile = fopen("myinput.txt","r");
        if (infile == NULL){
                 printf("Cannot open myinput.txt for reading \n");
                system("pause");
                exit(1);
        }

       while ( fscanf(infile,"%c",&c) != EOF ) {
        if (c != '\n'){
            total++;
            if (c>='0' && c<='9') numeric++;
            else if ((c>='a' && c<='z') ||(c>='A' && c<='Z')){
                 alpha++;
                 switch(c){
                 case 'a': case'A':
                 case 'e': case 'E':
                 case 'i': case 'I':
                 case 'o': case 'O':
                 case 'u': case 'U':
                     vowels++;
                 }
```

```
            }
        }
    }

    outfile = fopen("myoutput.txt","w");
    fprintf(outfile,"Number of vowel characters = %d \n",vowels);
    fprintf(outfile,"Number of numeric characters = %d \n",numeric);
    fprintf(outfile,"Number of alphabetic characters = %d \n",alpha);
    fprintf(outfile,"Total number of characters = %d \n",total);

    fclose(infile);
    fclose(outfile);

    return 0;
}
```

Question 5: (18 points)

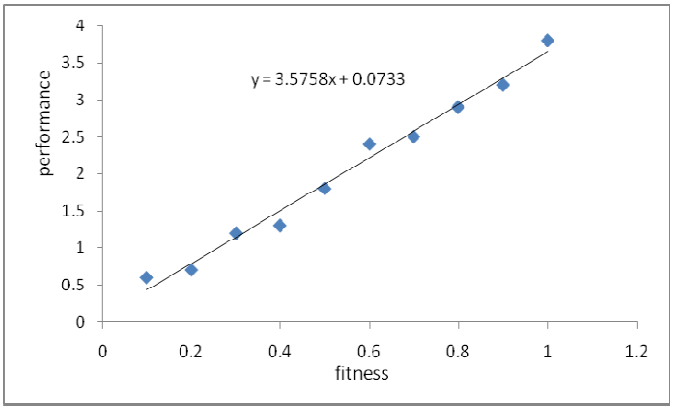
Given two sets of *n* observations **x** =[x_1, x_2, …, x_n] and **y** = [y_1, y_2, …, y_n] such that y_i corresponds to x_i where y_i is the i-th element in the set **y** and x_i is the i-th element in the set **x**. It is required to find the relationship between these two sets by curve fitting. A straight line can be defined by the following equation:

$y = mx + c$ where m and c are constants calculated as follows:

$$m = \frac{n\sum_{i=1}^{n} x_i y_i - (\sum_{i=1}^{n} x_i)(\sum_{i=1}^{n} y_i)}{n\sum_{i=1}^{n} x_i^2 - (\sum_{i=1}^{n} x_i)^2} \quad \text{and} \quad c = \frac{1}{n}(\sum_{i=1}^{n} y_i - m\sum_{i=1}^{n} x_i)$$

For example, one can study the relationship between student physical fitness and his academic performance using these formulas. Assume that there are 10 students with the data shown in the table below. The corresponding scatter diagram and straight line are as shown.

Fitness	Performance
0.1	0.6
0.2	0.7
0.3	1.2
0.4	1.3
0.5	1.8
0.6	2.4
0.7	2.5
0.8	2.9
0.9	3.2
1	3.8

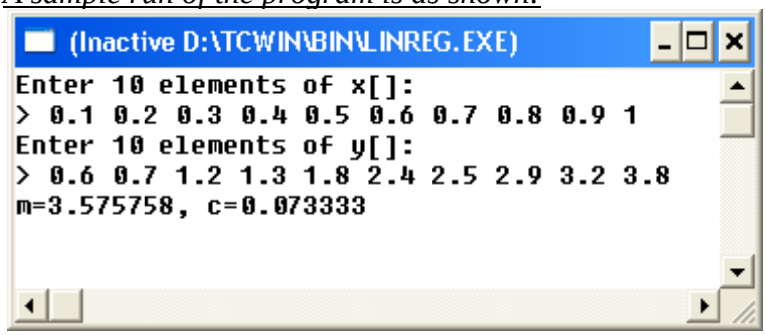


(Note that the data is not realistic; it is only given for the illustration of the curve fitting concept)

(i) Write a function **read_array** to read an array of type double.

(ii) Complete the main() function in the following program to read the data in the two arrays from the user, and then calculate and display m and c. Assume that the functions sum() and innerProduct() are predefined and should be used in your program.

A sample run of the program is as shown:




```
#include<stdio.h>
# define SIZE 10
// write read_array function here
    void read_array (double x[], int n){
            for (int i = 0;  i < n;  i++)
                    scanf("%lf", &x[i]);
    }
```

```
double sum(double x[], int n){
	double s=0;
	for (int i = 0;  i < n;  i++)
		s+=x[i];
	return s;
}

double innerProduct(double x[],double y[], int n){
	double p=0;
	for (int i = 0;  i < n;  i++)
		p+=x[i]*y[i];
	return p;
}

void main(void){
	double x[SIZE], y[SIZE];
	int i;

	printf("Enter %d elements of x[]: \n> ", SIZE);
	read_array (x, SIZE);

	printf("Enter %d elements of y[]: \n> ", SIZE);
	read_array (y, SIZE);

	double sumx=sum(x, SIZE);
	double sumy=sum(y, SIZE);

	double sumxy= innerProduct(x, y, SIZE);
	double sumxx= innerProduct(x, x, SIZE);

	double m = (SIZE*sumxy-sumx*sumy)/(SIZE*sumxx-sumx*sumx);
	double c  = (sumy-m*sumx)/SIZE;

	printf("m=%f, c=%f\n", m, c);

	system("pause");
	return 0;
}
```