I1101

Final

Question 1 - Program output *(20 pts - 20 minutes)*
Give the output on the screen of the following program (indicate a space using this symbol ␣):

```
#include<stdio.h>
double mystery(int A[], int i)
{
        i = A[i] * 2;
        A[0] = 0;
        return i+1;
}
int main()
{
        int i = 8, j = 5, k, A[] = {7,-2,0,1};
        double x = 0.005, y = -0.01;
        char c = 'c', d = 'd';  //ASCII value of 'd' is 100

        printf("#1# %d ##\n", (3*i-2*j) % (2*d-c));
        printf("#2# %d ##\n", 2*((i/5)+(4*(j-3))%(i+j-2)));
        k = 8;
        printf("#3# %d ##\n", ++k);
        k = 8;
        printf("#4# %d ##\n", k++);
        printf("#5# %9.4lf ##\n", ++x);
        printf("#6# %d ##\n", c>d);
        x = 0.005;
        printf("#7# %+9.2lf ##\n", 2*x+(y==0));
        printf("#8# %-9.2lf ##\n", 2 * x + y == 0);
        printf("#9# %d ##\n", (x>y) && (i>0) || (j<5));
        printf("#10# %d ##\n", i%=j);

        k = 1;
        for (i = 0; i < 5; i++)
        {
                for (j = 0; j < i; j++)
                {
                        if (i*j % 2) continue;
                        switch (i*j) {
                                case 0:
                                case 2:
                                case 4: k++;
                                case 6: break;
                                case 8: k--;
                                case 12:
                                default: k++;
                        }
                }
                printf("#%d# %d ##\n", 11 + i , k);
        }
        printf("#16# %0.0lf ##\n", mystery(A, 0));
        printf("#17# %d ##\n", A[0]);
        printf("#18# %d ##\n", A[2%4]);
        printf("#19# %d ##\n", A[1]%3);
        printf("#20# %d ##\n", -A[1]%3);
        return 0;
}
```

Solution

#1# 14 ##
#2# 18 ##
#3# 9 ##
#4# 8 ##
#5# ␣␣1.0050 ##
#6# 0 ##
#7# ␣␣␣␣+0.01 ##
#8# 0.01␣␣␣␣ ##
#9# 1 ##
#10# 3 ##
#11# 1 ##
#12# 2 ##
#13# 4 ##
#14# 5 ##
#15# 8 ##
#16# 15 ##
#17# 0 ##
#18# 0 ##
#19# −2 ##
#20# 2 ##

Grade distribution
Total: 20 points

- 1 pt for each correct line, 0 otherwise

Question 2 - Cost of a telephone call *(30 pts - 30 minutes)*

Write a program that asks the user to enter a number of minutes and a number of seconds, then calculates and displays the cost of a telephone call according to the following rules:

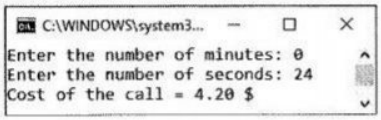
- a unit is equal to 20 seconds;
- the first minute is indivisible and is equal to 3 units.

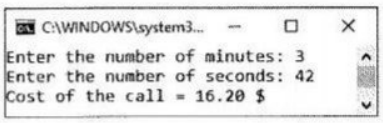
A unit is billed at 1.40 $ for the the first 3 minutes and 1.20 $ for the remaining minutes.

Useful hints !!

The number of minutes and seconds shouldn't be negative and shouldn't exceed sixty.
Calculate first the total number of units, making sure to take into account the different possibilities, and then calculate the total cost based on the total number of units.

Running examples:





Solution

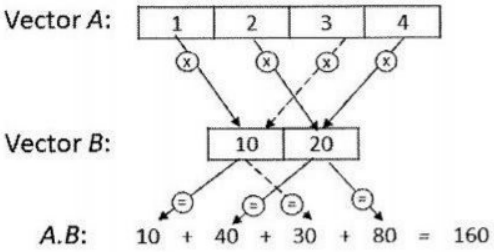
```
#include<stdio.h>
int main()
{
        int minutes, seconds, units, totalseconds;
        double total;

        do {
                printf("Enter the number of minutes: ");
                scanf("%d", &minutes);
        } while (minutes < 0 || minutes >60);

        do {
                printf("Enter the number of seconds: ");
                scanf("%d", &seconds);
        } while (seconds < 0 || seconds >60);

        if (minutes == 0)
                units = 3;
        else
        {
                totalseconds = minutes * 60 + seconds;
                units = totalseconds / 20;
                if (totalseconds % 20)
                        units++;
        }

        if (units >= 9)
                total = 9 * 1.4 + (units - 9)*1.2;
        else
                total = units*1.4;

        printf("Cost of the call = %.2lf $\n", total);

        return 0;
}
```

Grade distribution
Total: 30 points

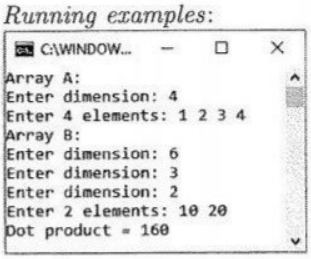
- 5 pts reading the number of minutes and seconds using a loop, 0 otherwise.
- 10 pts for calculating the toal number of units. Max 5 pts if there's a mistake. 0 otherwise.
- 10 pts for calculating the cost. Max 5 pts if there's a mistake. 0 otherwise.
- 5 pts for printing the value with 2 digits after the comma. 0 otherwise.

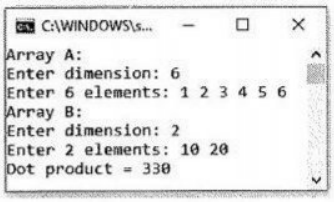
Question 3 - Dot product of two unequal-length vectors *(50 pts - 30 minutes)*

We are interested to perform the dot product for unequal-length vectors.
Algebraically, the dot product is the sum of the products of the corresponding entries of the two sequences of numbers. The dot product of two unequal-length vectors $A = [a_0, a_1, \cdots, a_{N-1}]$ and $B = [b_0, b_1, \cdots, b_{M-1}]$, where $N > 0$, $0 < M < N$ and $N = k \times M, k \in \mathbb{N} - \{0\}$, is defined as: $A \cdot B = a_0b_0 + \cdots + a_kb_{M-1} + a_{k+1}b_0 + \cdots + a_{N-1}b_{M-1}$.



Running examples:





1. Write the function `int READDIM_A()` that reads and returns a strictly positive integer N less than 50;
2. Write the function `int READDIM_B(int N)` that reads and returns a strictly positive integer M divisible by N;
3. Write the function `void READARRAY(int Arr[], int C)` that reads `C` positive integers of the array `Arr`;
4. Write the function `int DOTPRODUCT(int A[], int N, int B[], int M)` that returns the dot product of two unequal-length arrays;
5. **Using all the above written functions**, write a `main` function that reads two unequal-length arrays and then displays their dot product.

Solution

```
#include<stdio.h>
#define size 50

int READDIM_A()
{
        int N;
        do {
                printf("Enter dimension: ");
                scanf("%d", &N);
        } while (N <= 0 || N>size);
        return N;
}

int READDIM_B(int N)
{
        int M;
        do {
                printf("Enter dimension: ");
                scanf("%d", &M);
        } while (M <= 0 || M > N || N % M);
        return M;
}

void READARRAY(int Arr[], int C)
{
        int i;
        printf("Enter %d elements: ", C);
        for (i = 0; i < C; i++)
                scanf("%d", &Arr[i]);
}
```

```
int DOTPRODUCT(int A[], int N, int B[], int M)
{
        int i, j, sum = 0;
        for (i = 0, j = 0; i < N; i++, j++)
                sum += A[i] * B[j%M];
        return sum;
}

int main()
{
        int A[size], B[size];
        int N, M;

        printf("Array A:\n");
        N = READDIM_A();
        READARRAY(A, N);

        printf("Array B:\n");
        M = READDIM_B(N);
        READARRAY(B, M);

        printf("Dot product = %d\n", DOTPRODUCT(A,N,B,M));

        return 0;
}
```

Grade distribution
Total: 50 points

- 5 pts for `READDIM_A`. 0 if there's mistakes.
- 10 pts for `READDIM_B`. -5 pts if there's a mistake in the modulo operator. 0 for other mistakes.
- 5 pts for `READARRAY`. 0 if there's mistakes.
- 20 pts for `DOTPRODUCT`. -10 pts if there's a mistake in the modulo operator. -10 pts if using more than one loop. 0 for other mistakes.
- 10 pts for the `main` function. 0 if there's a mistake.

Good Luck !

Question 1 - Program output *(11 pts - 5 minutes)*
Give the output on the screen of the following program (indicate a space using this symbol ␣):

```
#include<stdio.h>
int main()
{
        int a = 7, b = 4, c;
        double x = 0.02, y = -0.05, z = 0.1 ;
        printf("#1# %d ##\n", (a + b) % 1234);
        printf("#2# %d ##\n", a + b + 1 % 1234);
        printf("#3# %d ##\n",  4*((a/6)+(4*(b-2))%(a-b)));
        a = 7;
        printf("#4# %d ##\n", ++a);
        a = 7;
        printf("#5# %d ##\n", a++);
        printf("#6# %9.4lf ##\n", ++x);
        printf("#7# %d ##\n", a>b);
        x=0.02;
        printf("#8# %+9.2lf ##\n", 2 * x + (y == 0));
        printf("#9# %+-9.2lf ##\n", 2 * x + y);
        printf("#10# %d ##\n", (x>y) && (a<0) || (b>5));
        a = 7;
        printf("#11# %d ##\n", a%=b);
        return 0;
}
```

Solution

#1# 11 ##
#2# 12 ##
#3# 12 ##
#4# 8 ##
#5# 7 ##
#6# ␣␣1.0200 ##
#7# 1 ##
#8# ␣␣␣+0.04 ##
#9# -0.01␣␣␣ ##
#10# 0 ##
#11# 3 ##

Grade distribution
Total: 11 points

- 1 pt for each correct line, 0 otherwise

Question 2 - Program output *(9 pts - 15 minutes)*

Give the output on the screen of the following program (indicate a space using this symbol ␣):

```
#include<stdio.h>
double mystery(int T[], int i)
{
        int k;
        for(k=0 ; k<i ; k++)
                T[k] -= T[i] - k;
        return T[k] + 1;
}
int main()
{
        int a = 7, b = 4, c, T[] = {4,-4,8,0};
        double x = 0.02, y = -0.05, z = 0.1 ;
        c = 1;
        for (a = 0; a < 5; a++)
        {
                for (b = a; b < 5; b++)
                        switch (a-b) {
                                case 0:
                                case 2:
                                case 4: c++;
                                case 6: break;
                                case 8: c--;
                                case 12:
                                default: c++;
                        }
                printf("#%d# %d ##\n", 12 + a , c);
        }
        printf("#17# %0.01f ##\n", mystery(T, 2));
        printf("#18# %d ##\n", T[0]);
        printf("#19# %d ##\n", T[0%4]);
        printf("#20# %d ##\n", -T[0]%3);
        return 0;
}
```

Solution

#12# 6 ##
#13# 10 ##
#14# 13 ##
#15# 15 ##
#16# 16 ##
#17# 9 ##
#18# -4 ##
#19# -4 ##
#20# 1 ##

Grade distribution
Total: 9 points

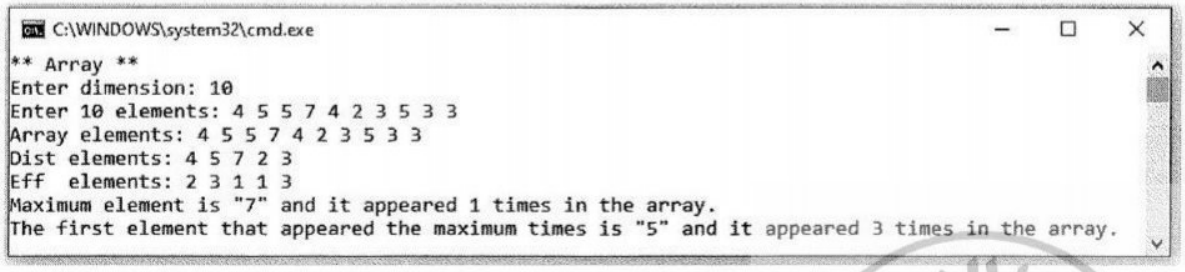
- 1 pt for each correct line, 0 otherwise

Question 3 - Array manipulation using functions (*80 pts - 50 minutes*)

1. Write the function `int READDIM()` that reads and returns a strictly positive integer N less than 50;
2. Write the function `void READARRAY(int T[], int N)` that fills `N` positive integers in the array `T`;
3. Write the function `void PRINTARRAY(int T[], int N)` that prints `N` elements of the array `T`;
4. Write the function `int INDEX_MAX(int T[], int N)` that returns the **index** of the first occurrence of the maximum element appearing in `T`;
5. Write the function `int NB_OCC(int T[], int N, int dist[], int eff[])` that given an array `T` containing `N` elements, performs the following:
   - Fills the array `dist` with the distinct elements of `T`;
   - Fills the array `eff` with the number of occurrence of each distinct element of `T`;
   - Returns the number of distinct elements of `T`.

   Example:
   Let `T` contains the following elements: 4 5 5 7 4 2 3 5 3 3.
   - Array `dist` will contain the following elements: 4 5 7 2 3.
   - Array `eff` will contain the following elements: 2 3 1 1 3, since the elements 4 appears 2 times in `T`, the elements 5 appears 3 times in `T`, and so on $\cdots$
   - The returned value is 5 since the number of distinct elements is 5 in `T`.
6. **By calling all the above written functions**, write a `main` function that performs the following:
   - Read the dimension of an array `T`;
   - Fills `T` with elements;
   - Displays the elements of `T`;
   - Fills and displays arrays `dist` and `eff`;
   - Calculates and displays the maximum element and its number of occurrence in `T`;
   - Calculates and displays the first element that appeared the maximum times and its number of occurrence in `T`.

Running example:



Solution

```
#include<stdio.h>
#define size 50

int READDIM()
{
        int N;
        do {
                printf("Enter dimension: ");
                scanf("%d", &N);
        } while (N <= 0 || N>size);
        return N;
}

void READARRAY(int Arr[], int C)
{
        int i;
        printf("Enter %d elements: ", C);
        for (i = 0; i < C; i++)
                scanf("%d", &Arr[i]);
}

void PRINTARRAY(int Arr[], int C)
{
        int i;
        for (i = 0; i < C; i++)
                printf("%d ", Arr[i]);
        printf("\n");
}

int index_max(int A[], int N)
{
        int i , imax=0;
        for(i=1 ; i <N; i++)
                if(A[i]>A[imax])
                        imax = i;
        return imax;
}

int Nb_OCC(int A[], int N, int dist[], int eff[])
{
        int i, j, c = 0;
        for (i = 0; i < N; i++)
        {
            for(j=0; j<c; j++)
                if(dist[j]==A[i])
                {
                    eff[j]++;
                    break;
                }
            if(j>=c)
            {
                dist[j]=A[i];
                eff[j]=1;
                c++;
            }
        }
        return c;
```

```
}


int main()
{
        int A[size], dist[size], eff[size];
        int N, M , imax;

        printf("** Array **\n");
        N = READDIM();
        READARRAY(A, N);
        printf("Array elements: ");
        PRINTARRAY(A,N);

        M = Nb_OCC(A, N, dist , eff);
        printf("Dist elements: ");
        PRINTARRAY(dist ,M);
        printf("Eff  elements: ");
        PRINTARRAY(eff ,M);

        imax = index_max(dist , M);

        printf("Maximum element is \"%d\" and it appeared %d
            times in the array.\n",dist[imax],eff[imax]);


        imax = index_max(eff , M);

        printf("The first element that appeared the maximum times
            is \"%d\" and it appeared %d times in the array.
            \n",dist[imax],eff[imax]);

        return 0;
}
```

Grade distribution
Total: 80 points

1. `READDIM` over 5 pts:
   - 5 pts if it's correct
   - -3 pts if `while` is missing or wrong `while` condition
   - -3 pts if `return` statement is missing or wrong calculated return value
   - -3 pts if there's a mistake in `scanf`
2. `READARRAY` over 5 pts:
   - 5 pts if it's correct
   - -3 pts if `for` loop is missing or wrong `for` condition or wrong syntax
   - -3 pts if there's a `return` statement
   - -3 pts if there's a mistake in `scanf`
3. `PRINTARRAY` over 5 pts:
   - 5 pts if it's correct
   - -3 pts if `for` loop is missing or wrong `for` condition or wrong syntax
   - -3 pts if there's a `return` statement
   - -3 pts if there's a mistake in `printf`

4. `INDEX_MAX` over 15 pts:
   - 15 pts if it's correct
   - -10 pts if `for` loop is missing or wrong `for` condition or wrong syntax
   - -5 pts if `return` statement is missing
   - -5 pts if there's a mistake in the `if` statement
   - -5 pts if not using strict comparison (>= is considered wrong)
   - -10 pts if working on array values not indices
5. `NB_OCC` over 30 pts:
   - 30 pts if it's correct
   - -15 pts if not using 2 nested `for` loops correctly
   - -10 pts if `return` statement is missing or wrong calculated return value
   - -10 pts if there's a mistake in filling the array `dist`
   - -10 pts if there's a mistake in filling the array `eff`
6. `main` over 20 pts:
   - 20 pts if it's correct
   - -5 pts if calling `READDIM` or `READARRAY` or `PRINTARRAY` incorrectly
   - -5 pts if calling `NB_OCC` incorrectly
   - -10 pts if calling `INDEX_MAX` incorrectly in any of the 2 calls

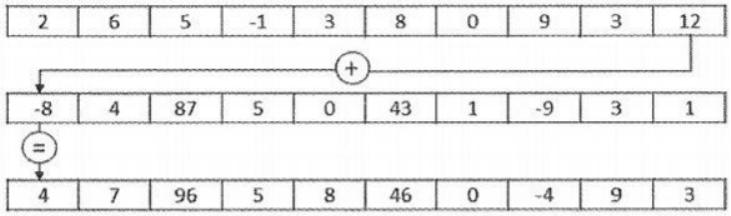
Good Luck !

Question 1

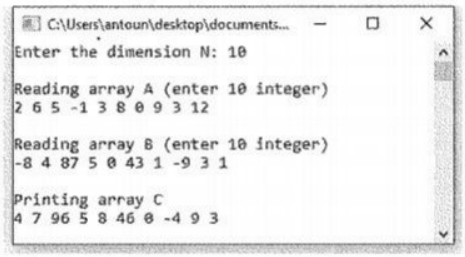
Write a program that:

1. reads the effective dimension `N` of an array (maximum dimension: 50 components);
2. then fills two arrays `A` and `B` each with `N` integers entered on the keyboard;
3. then fills a third array `C` by the inverse summation of `A` and `B`;
4. then displays array `C`.

PS: the inverse summation can be calculated for each element of the array as illustrated below:



Running example:



Solution

```
#include<stdio.h>
#define SIZE 50

void main()
{

        int N;
        int A[SIZE], B[SIZE],C[SIZE];
        int i, j;

        // reading the dimension
        do {
                printf("Enter the dimension N: ");
                scanf("%d", &N);
        } while (N<0 || N>SIZE);


        printf("\nReading array A (enter %d integer)\n",N);
        for (i = 0; i < N; i++)
                scanf("%d", &A[i]);

        printf("\nReading array B (enter %d integer)\n", N);
        for (i = 0; i < N; i++)
                scanf("%d", &B[i]);
```

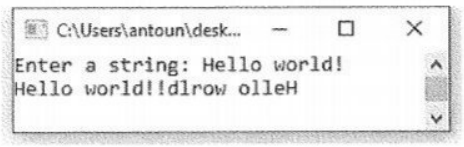
```
    for (i = 0, j = N - 1; i < N; i++,j--)
        C[i] = A[j] + B[i];

    printf("\nPrinting array C\n");
    for (i = 0; i < N; i++)
        printf("%d ", C[i]);
}
```

Grade distribution
Total: 20 points

- 4 pts for reading the dimension
- 4 pts for reading both arrays
- 8 pts for calculating the inverse summation
- 4 pts for printing the array

Question 2

Write a program that reads a string STR from the keyboard and then adds the reverse of STR to its end (i.e. STR should contain the original string and its reverse).
You may assume that the size of the string STR is big enough to hold the string and its reverse.
The usage of any function of the string.h library is not allowed.
Running example:



Solution

```
#include <stdio.h>

void main()
{
        char STR[100];
        int i , j;

        printf("Enter a string: ");
        gets(STR);

        i = 0;
        // loop until the \0
        while (STR[i]) // go to the end of STR
                i++;
        // j goes backward from i-1 till 0
        j = i - 1;

        while (j >= 0) // copy STR to the end of STR
        {
                STR[i] = STR[j];
                i++;
                j--;
        }
        // end of string
        STR[i] = '\0';

        puts(STR);
}
```

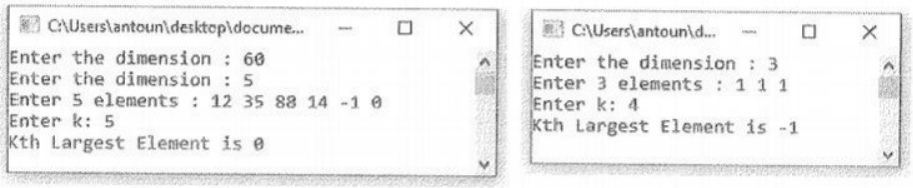
Grade distribution
Total: 20 points

- 2 pts for reading the string
- 2 pts for reaching the end of the string
- 2 pts for creating a new index j = i-1
- 4 pts for the loop
- 2 pts for the incrementation and the decrementation of both counters
- 4 pts for STR[i] = STR[j];
- 2 pts for putting \0 at the end of the string
- 2 pts for printing the string

Question 3

1. Write the function `int READDIM()` that reads and returns a strictly positive integer less than 50;
2. Write the function `void READARRAY(int Arr[], int N)` that reads `N` positive integers of the array `Arr`;
3. Write the function `void SORT(int Arr[], int N)` that sorts in increasing order the array `Arr`;
4. Write the function `int Kth(int Arr[], int N, int k)` that returns
   - the kth element of the array `Arr` if it exists ;
   - -1 if it doesn't;
5. **Using all the above written functions**, write a `main` function that displays the k^{th} **largest** element in an array (maximum size 50) of positive integers.

Running example:



Solution

```
#include<stdio.h>
#include<conio.h>
#define SIZE 50

int READDIM()
{
        int n;
        do
        {
                printf("Enter the dimension : ");
                scanf("%d", &n);
        } while (n <=0 || n>SIZE);
        return n;
}

void READARRAY(int Arr[], int dim)
{
        int i;
        printf("Enter %d elements : ", dim);
        for (i = 0; i<dim; i++)
                do {
                        scanf("%d", &Arr[i]);
                } while (Arr[i] < 0);
}

void SORT(int Arr[], int dim)
{
        int i, j, temp;
        for (i = 0; i < dim; i++) {
                for (j = i + 1; j < dim; j++) {
                        if (Arr[i] > Arr[j]) {
                                temp = Arr[i];
```

```
                                Arr[i] = Arr[j];
                                Arr[j] = temp;
                        }
                }
        }
}

int Kth(int Arr[], int dim, int k)
{
        if(k>0 && k<=dim)
                return Arr[k-1];
        return -1;
}

void main()
{
        int n, T[SIZE],   k;

        n = READDIM();
        READARRAY(T, n);
        SORT(T, n);

        printf("Enter k: ");
        scanf("%d", &k);

        printf("Kth Largest Element is %d\n", Kth(T,n, n-k+1));
        getch();
}
```

Grade distribution
Total: 30 points

- 2 pts for READDIM
- 5 pts for READARRAY
- 10 pts for SORT
- 6 pts for Kth
- 7 pts for main:
  - 1 pt for calling READDIM
  - 1 pt for calling READARRAY
  - 1 pt for calling SORT
  - 1 pt for calling Kth
  - 2 pts for the parameters of kth function
  - 1 pt for printf

Good Luck !

I1101
Algorithms and Programming

Problem I — 22 points

1. Write a function **isPrime** that takes an integer *a*, returns 1 if *a* is prime and returns 0 if *a* is not prime. **[6 points]**

```
int isPrime(int a){ // signature: 1 point
    int i;
    for(i = 2 ;  i <= a/2 ; i++) // loop: 2 point
        if( a%i == 0) // if + condition : 1 point
            return 0; // 1 point
    return 1; // 1 point
}
```

2. Two integers *a* and *b* are said to be **coPrime** if the only positive integer that divides both is 1. Write a function **areCoPrime** that takes two integers *a* and *b*, returns 1 if *a* and *b* are coprime, 0 if not. **[8 points]**

```
int areCoPrime(int a, int b){ // signature  1 point
    int i;

    // loop + condition : 3 points
    for(i = 2; i <= a && i <= b ; i ++)
        if (a%i == 0 && b%i == 0 ) // if + condition : 2 points
            return 0;  // 1 point
    return 1; // 1 point
}
```

3. Write a main program that asks the user to enter two integers *a* and *b* and determines if *a* and *b* are primes or not, coprime or not. **[8 points]**

```
void main(){
    // declaration + data entry (following the format) : 2 points
    int a, b;
    printf("Enter two numbers:");
    scanf("%d%d",&a,&b);

    // 2 points
    if(isPrime(a) == 1)
        printf("%d is prime\n",a);
    else
        printf("%d is not prime\n",a);
    // 2 points
    if(isPrime(b) == 1)
        printf("%d is prime\n",b);
    else
        printf("%d is not prime\n",b);
```

```
    // 2 points
     if(areCoPrime(a,b) == 1)
          printf("%d and %d are coprime\n",a, b);
     else
          printf("%d and %d are not coprime\n",a, b);

}
```

Problem II — 31 points

In this exercise we consider strings denoting complete file names. A complete file name consists of 2 parts: a name and an extension (example: prog.c is a file name, where "prog" is the name, and "c" is the extension).
Therefore, a file name is valid if it satisfies the following conditions:

- It starts with an alphabetical character
- It contains one and only one '.' (dot character), that comes between the name and the extension (it does not appear at the beginning or at the end of the file name).

1. Write the function **validName** that takes a string and tests whether it is a valid name or not. **[6 points]**

```
int validName(char n[]){ // signature : 1 point
     int i, c;
     // check if the string begins with a letter : 1 point
     if( !((n[0] >= 'a' && n[0] <='z') || (n[0] >= 'A' && n[0] <='Z'))
)
          return 0;


     // check if the string contains one and only one dot :  2 points
     c = 0;
     for(i = 0; i<strlen(n); i ++){

          if(n[i] == '.')
               c ++;
     }

     if(c != 1)
          return 0;

     // check if the dot comes between the name and the ext: 1 point
     if (n[0] == '.' || n[strlen(n) -1 ] == '.')
          return 0;

     return 1; // valid cases : 1 point
}
```

2. Write the function **split** that takes a file name as string (supposed to be valid) and generates the name and the extensions in two other strings. **[9 points]**

```
void split(char f[], char n[], char e[]){ // signature : 1 point

   // declaration + initialization : 1 point
   int i, j, k;
   i = j = k = 0;


   while(f[i] != '.') // loop + condition : 1 point
         // get the value : 0.5 point
         // increment the indices : 1 point
         n[j++] = f[i++];
   n[j] = '\0'; // close the string :  0.5 point

   i++; // jump one char : 1 point

   while(f[i] != '\0') // loop + condition :  1 point
         // get the value : 0.5 point
         // increment the indices : 1 point
         e[k++] = f[i++];
   e[k] = '\0'; // close the string :  0.5 point
}
```

3. Write a C void function **extensionType** that takes an extension and prints on the screen the file type according to the following listed extension. **[6 points]**

Extension	Type of the file
"c"	A C program file
"docx"	A document file
"avi"	A video file
Otherwise	Unknown tyoe

PS: you may use the function strcmp(char []s1, char []s2) of the string.h library. This function returns 0 if the two strings s1 and s2 are the same,

```
void extensionType(char e[]){ // signature : 1 point

      // correct use of strcmp in the condition : 2 points
      // correct if/else structure : 2 points
      // output : 1 point

      if(strcmp(e,"c") == 0)
            printf("C program\n");
      else if(strcmp(e,"docx") == 0)
            printf("document\n");
      else if(strcmp(e,"avi") == 0)
            printf("video\n");
      else printf("unkown\n");
}
```

4. Write a main function that reads a string (of Max 30 characters) then invokes the **validName** function. In the case of an invalid name it prints "Invalid" otherwise it prints the name, the extension and the type of the file. **[10 points]**

Output examples (next page):

Execution1	Execution2	Execution3
Enter file name? 1File1.doc Invalid	Enter filename? File1.docx Name: file1 Extension: doc The file is a document file.	Enter string? Prog.c Name: Prog Extension: c The file is a C program file.

```
void main(){
      char fn[30], n[30], e[30]; // declaration : 1 point
      gets(fn); // 1 point

      if(validName(fn) == 1 ){ // 1 point
            split(fn,n,e); // 2 points
            puts(n); // 0.5 point
            puts(e); // 0.5 point
            extensionType(e); // 2 points
      }
      // 2 points
      else
            printf("invalid\n");

}
```

Lab Part

Problem III — **5 points**

```
// 5 points if complete correct solution
// 3 points if the solution is not correct, but it is composed of a
sequence of "F" (any wrong number) followed by exactly one "G"
// 0 point otherwise
```

```
F
F
F
F
F
G
```

Problem IV **10 points**

1. Write the function **perfect** that checks if a number is perfect or not. The function returns 1 if the number is perfect and 0 otherwise. Note that an integer N is considered perfect if the sum of the divisors of N, excluding the number N itself, is equal to the number N.
   Example: 6 is perfect (because 1+2+3 = 6) **[3 points]**

```
int perfect (int x){
      int s = 0, i; // 0.5 point
      for(i = 1; i <= x/2; i++) // loop   : 0.5 point
            if(x%i == 0) // divisor : 0.5 point
                  s+=i;     // sum : 0.5 point
      // if + return : 1 point
      if(s==x)
            return 1;
      return 0;
}
```

2. Write the function **ReadArray** that allows to read the value of an array of size M. All the elements of the array should be between 10 and 100. **[3 points]**

```
void ReadArray(int X[], int M){
   int i;
   for(i =0 ; i < M ; i++){ // loop : 1 point
         //do/while 0.5 point + correct condition 1 point
         do{
               scanf("%d", &X[i]); // read the value : 0.5 point
         }while(X[i] < 10 || X[i] > 100);
   }
}
```

3. Write a main program that declares an array A of maximum size 20. The program should prompt the user for the effective size of A (should be positive), and read its values using the function **ReadArray**. Then the program should count and print the number of perfect numbers in A. **[4 points]**

```
void main(){
      int A[20], M, i,  c = 0; // declaration  : 0.5 point
      // read M  1 point
      do{
            scanf("%d",&M);
      }while(M<=0);

      ReadArray(A,M);  // calling the function 0.5 point

      for(i = 0 ; i < M ; i ++) // 0,5 point
            if(perfect(A[i]) == 1) // 0.5 point
                  c++;                // 0.5 point
```

```
        printf("%d",c);  // 0.5 point
}
```

Université Libanaise
Faculté des Sciences I

2ème Session 2017/2018
Durée : 2 heures

I1101
Algorithme et Programmation

Problème I — 27 points

1. Ecrire une fonction C ***AfficheTableau*** qui prend en paramètres un tableau ainsi que deux indices *m* et *n* et qui affiche les éléments du tableau entre ces deux indices. **[4 points]**
2. Ecrire une fonction C ***MaxMin*** qui prend en paramètres un tableau d'entiers ainsi que sa taille puis calcule et affiche (en utilisant *AfficheTableau*) le maximum de la première moitié du tableau, et calcule et affiche le minimum de la deuxième moitié comme dans l'exemple d'exécution. **[11 points]**
3. Ecrire une fonction C ***LireTableau*** qui prend en paramètre un tableau et sa taille puis remplit le tableau par des valeurs entrées par l'utilisateur. **[4 points]**
4. Ecrire une fonction ***main*** qui demande à l'utilisateur la taille du tableau (doit être un nombre pair), remplit le tableau (fonction *LireTableau*), puis calcule le maximum de la première moitié et le minimum de la deuxième moitié (*MaxMin*). **[8 points]**

Exemple d'exécution :

Donner la taille du tableau (nb pair) : **9**
Donner la taille du tableau (nb pair) : **10**
Entrer les 10 éléments du tableau : **5 2 8 3 1 10 9 7 18 12**
Le maximum de la première moitié du tableau 5 2 8 3 1 est 8
Le minimum de la deuxième moitié du tableau 10 9 7 18 12 est 7

Problème II — 27 points

1. Ecrire une fonction ***SeulementMajuscules*** qui prend en paramètre une chaîne de caractères S et vérifie si elle ne contient que des majuscules. La fonction renvoie 1 si la chaîne S contient uniquement des majuscules 0 sinon. **[6 points]**
2. Ecrire une fonction ***CompteLettresMajuscules*** qui prend une chaîne de caractère S et un tableau T (de taille fixe 26) et compte dans T le nombre d'occurrence de chaque lettre majuscule trouvée dans S. On suppose que le premier élément de T contienne le nombre d'occurrence de A, le deuxième élément contienne le nombre d'occurrence de B, et ainsi de suite ... **[5 points]**

   Exemple : Si la chaîne S est "AMBMABDAMCMZM", le tableau T est :

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
3	2	1	1	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	1

3. Ecrire un programme main qui :
   - Demande à l'utilisateur d'entrer deux chaînes S1 et S2 en s'assurant que chacune contient exclusivement des lettres majuscules (en utilisant la fonction *SeulementMajuscules*). **[4 points]**
   - Le programme doit alors appeler la fonction *CompteLettresMauscules* pour remplir dans les deux tableaux *T1* et *T2* le nombre d'occurrence de chaque lettre majuscule dans les deux chaînes *S1* et *S2*, puis calculer (dans un différent tableau T) le nombre d'occurrence de chaque lettre dans les deux chaînes *S1* et *S2*. **[5 points]**
   - Enfin, le programme doit utiliser les deux tableaux *T1* et *T2* pour vérifier si la chaine *S1* est une sous-chaine de *S2* (toutes les lettres de *S1* apparaissent dans S2). **[6 points]**

Partie Lab

Problème III — 4 points

Quelles sont les valeurs des variables « i » et « count » à la fin de ce bloc de programme.

```
int count = 0;
int i = 0;
while(count < 20){
        i ++;
        if(count % 5 == 0)
                break;
        if(count % 2 == 0)
                count += 2;
        if (count % 4 == 0)
                count--;
        if (count % 3 == 0)
                count *= 2;
}
```

Problème IV — 11 points

1. Ecrire une fonction qui prend en paramètre un entier *n* et retourne son plus grand chiffre.
   Exemple 1 : si n = 3091, votre fonction doit retourner 9
   Exemple 2 : si n = 1385201, votre fonction doit retourner 8

2. Ecrire un programme main qui demande à l'utilisateur d'entrer un entier positif *n*, puis calcule et affiche son plus grand chiffre.

I1101
Algorithms and Programming

Problem I — 27 points

1. Write a C function ***DisplayArray*** that takes as parameter an array and two index *m* and *n*. This function should display all arrays' elements that are between these indexes. **[4 points]**
2. Write a C function ***MaxMin*** that takes as parameter an array of integers and its size that calculates and displays (using *DisplayArray*) the maximum of first half of array, and calculates and displays the minimum of second half of array as in the example of execution. **[11 points]**
3. Write a C function **ReadArray** that take as parameter an array and its size then fills the array by values entered by the user. **[4 points]**
4. Write a ***main*** function that asks the user to enter the size of the array (should be an even number), fills the array (function *ReadArray*), then calculates the maximum of first half and the minimum of second half of the array (*MaxMin*). **[8 points]**

Example of execution:

Enter the size of the array (even nb): **9**
Enter the size of the array (even nb): **10**
Enter 10 elements of array: **5 2 8 3 1 10 9 7 18 12**
The maximum of first half of array 5 2 8 3 1 is 8
The minimum of second half of array 10 9 7 18 12 is 7

```
# include<stdio.h>
#define taille 50
void AfficheTableau(int T[], int m, int n)
{
        int i;
        for (i=m; i<n; i++)
                printf("%d ",T[i]);
}

void MaxMin(int T[], int n)
{
        int i, max=T[0],min=T[n/2];
        for(i=0;i<n/2;i++)
        {
                if(T[i]>max) max=T[i];
                if(T[n/2+i]<min) min=T[n/2+i];
        }
        printf("Le maximum de la premiere partie du tableau ");
        AfficheTableau(T,0,n/2);
        printf(" est %d\n",max);
        printf("Le minimum de la deuxieme partie du tableau ");
        AfficheTableau(T,n/2,n);
        printf(" est %d\n",min);
}
```

```
void LireTableau(int T[], int n)
{
        int i;
        printf("Entrer les %d elements du tableau : ",n);
        for(i=0;i<n;i++)
                scanf("%d",&T[i]);
}
main ()
{
        int n, T[taille];
        do
        {
                printf("Donner la taille du tableau (nb pair) : ");
                scanf("%d",&n);
        } while (n%2 !=0);
        LireTableau(T,n);
        MaxMin(T,n);
}
```

Problem II **26 points**

1. Write the function ***OnlyCapitals*** that takes a string S and checks if it contains only capital letters. The function returns 1 if the *S* contains only capital letters, 0 otherwise. **[6 points]**
2. Write the function ***CountCapitalLetters*** that takes a string S and an array T (of fixed size 26) and counts in *T* the number of occurrence of each capital letter found in *S*. We suppose that we put in the first element of *T* the number of occurrence of A, in the second element of *T* the number of occurrence of B, and So on... **[5 points]**

   Example: if the string *S* is "AMBMABDAMCMZM", the array *T* is:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
3	2	1	1	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	1

3. Write a main program that:
   - Asks the user to enter two strings *S1* and *S2* by making sure that each of them contains exclusively capital letters (using the function *OnlyCapital*). **[4 points]**
   - The program should then call the function *CountCapitalLetters* to get in arrays *T1* and *T2*, the number of occurrence of each capital letter in strings *S1* and *S2* then compute (in a different array *T*) the number of occurrence of each letter in strings *S1* and *S2*. **[5 points]**
   - Finally the program should use the two arrays *T1* and *T2*, to check if the String *S1* is a substring of *S2* (all the letters of *S1* appears in *S2*). **[6 points]**

```
# include<stdio.h>
#include<string.h>
#define size 50
#define N 26
int OnlyCapitals(char S[])
{
```

```
	int i;
	for(i=0;S[i]!='\0';i++) if(S[i]<'A' || S[i]>'Z') return 0;
	return 1;
}
void CountCapitalLetters(char S[], int T[])
{
	int i;
	for(i=0;S[i]!='\0';i++)
	{
		T[S[i]-'A']++;
	}
}
main ()
{
	int i,j;
	char s1[size],s2[size];
	int i,j,T1[N]={0},T2[N]={0},sub;
	do
	{
		printf("Enter s1: ");
		gets(s1);
	} while(OnlyCapitals(s1)==0);
	do
	{
		printf("Enter s2: ");
		gets(s2);
	} while(OnlyCapitals(s2)==0);
	CountCapitalLetters(s1,T1);
	CountCapitalLetters(s2,T2);
	for(i=0;i<N;i++) printf("%d ",T1[i]);
	printf("\n");
	for(i=0;i<N;i++) printf("%d ",T2[i]);
	printf("\n");
	sub=1;
	for(i=0;i<N;i++)
		if(T1[i]>T2[i])
		{
			sub=0;
			break;
		}
	if(sub) printf("%s is a substring of %s\n",s1,s2);
	else printf("Not substring\n");
}
```

Lab Part

Problem III — 4 points

What is the value of the variables "i" and "counts" at the end of this program block?

```
int count = 0;
int i = 0;
while(count < 20){
        i ++;
        if(count % 5 == 0)
                break;
        if(count % 2 == 0)
                count += 2;
        if (count % 4 == 0)
                count--;
        if (count % 3 == 0)
                count *= 2;
}
```

Problem IV — 11 points

1. Write a function that takes as parameter an integer *n* and returns the largest digit in *n*.
   Example 1: If n = 3091, your function must return 9
   Example 2: If n = 1385201, your function must return 8

2. Write a main program that asks the user to enter a positive integer number *n*. Then find and print out the largest digit in *n*.

Lebanese University
Faculty of Sciences I

Final 2016/2017
Duration : 1 hour

I1101
Algorithms and Programming

Problem I — 30 points

We consider the following sequence:

$$\begin{cases} U_0 = 1 \\ U_n = U_{n-1} + \dfrac{U_{n-1}}{n!} \end{cases}$$

1. Write the function **int Factorial(int n)** that returns the factorial of an integer *n*. **(9 points)**
2. Write the function **float Sequence(int n)** that returns the value of the element U_n in the previous sequence, and that uses the function Factorial. **(14 points)**
3. Write a main function that asks the user to enter a number *n*, calculates and prints the value of U_n. **(8 points)**

Problem II — 35 points

Two numbers *N1* and *N2* are said to be *brothers*, if each digit of *N1* appears at least one time in *N2*, and vise versa, each digit of *N2* appears at least one time in *N1*.
Examples:

- *N1 = 1164 and N2 = 612, are brothers.*
- *N1 = 905 and N2 = 9059, are bothers.*
- *N1 = 405 and N2 = 554, are not bothers.*

1. Write the function **Init** that initializes to -1 the value of an array's integer elements.
2. Write the function **Belongs** that checks if an integer value v belongs to an array of integers *T*. The function returns 1 if v belongs to *T* and 0 otherwise.
3. Write the function **Decompose** that takes an integer *n* and an array of integers *T*, and stocks the digits of *n* in *T*.
4. Using the previous functions, write a main function that prompts the user to enter the values of two integers *N1* and *N2*, and says if they are brothers or not.

Problem III — 35 points

Lab Question — 15 points

1. Write the function **isAlphabet** that checks if a character is an alphabet or not (the function returns 1 if the character is an alphabet, or 0 otherwise)
2. Write the function **EliminationOfNoAlphabet** that takes a string *S* and eliminates the non alphabet from it.

   Example: *if S = "ab A 132?CD#"*
   after elimination, the string S should become "abACD"

3. Write a function that takes two strings *S1* and *S2*, and regroups all the alphabets of the two strings (S1 and S2) in S1, and the other characters of both strings in S2.
   Example: *if S1 = "ab123??" and S2 = "&@Mp9C"*
   After calling the function they should become: S1="abMpC" and S2 = "123??&@9"

```
}

void main(){
      int A[20],B[20],i,j,n1,n2;
      do{
            printf("Enter two numbers: ");
            scanf("%d %d",&n1,&n2);
      }while(n1<0||n2<0);
      Int(A,20);
      Int(B,20);
      if(n1>=n2){
      Decompose(A,20,n1);
      Decompose(B,20,n2);
      }
      else{
            Decompose(A,20,n2);
            Decompose(B,20,n1);
      }
      for(i=0;A[i]!=-1;i++){
            for(j=0;B[j]!=-1;j++){
                  if(A[i]==B[j]){
                        break;
                  }
            }
            if(B[j]==-1){
                  printf("%d and %d are not brothers",n1,n2);
                  break;
            }
      }
      if(A[i]==-1)
            printf("%d and %d are brothers",n1,n2);
      getch();
}
```

Ex.3

```
int isAlphabet(char c){
      if((c>='a'&&c<='z')||(c>='A'&&c<='Z'))
            return 1;
      return 0;
}

void EliminationOfNoAlphabet(char A[]){

      int i,j,count;

      for(i=0;i<strlen(A);i++)
            if(!isAlphabet(A[i])){
                  for(j=i;j<strlen(A);j++){
                        A[j]=A[j+1];
                  }
                  i--;
            }
}
```

```
void Switch(char A[],char B[]){
	char S1[40],S2[40];
	int i,j=0,k=0;
	for(i=0;A[i]!='\0';i++){
		if(isAlphabet(A[i])){
			S1[j]=A[i];
			j++;
		}
		else{
			S2[k]=A[i];
			k++;
		}
	}

	for(i=0;B[i]!='\0';i++){
		if(isAlphabet(B[i])){
			S1[j]=B[i];
			j++;
		}
		else{
			S2[k]=B[i];
			k++;
		}
	}
	S1[j]='\0';
	S2[k]='\0';
	strcpy(A,S1);
	strcpy(B,S2);
}
```

Lebanese University
Faculty of Sciences I

Second session 2016/2017
Duration : 2 hours

I1101
Algorithms and Programming

Problem I — 10 points

Consider the following sequence:

$$U_0 = U_1 = 1; \qquad U_n = 2 \times U_{n-1} + 3 \times U_{n-2}$$

1. Write the function **int Seq (int n)** that returns the value of the element U_n.
2. Write a main function that print the first 20 elements of the sequence.

Problem II — 10 points

Write program that asks the user to enter a positive integer f then prints the factorial inverse a (fact-inverse) of f if $f = a!$. Example: the fact-inverse of 24 is 4 (as 24 = 4!), the fact-inverse of 120 is 5 (as 120 = 5!). If the number is not a factorial, the program prints the message "value not found".

Examples of execution:

Enter a value? 120
The fact-inverse is 5

Enter a value? 9
Value not found

Problem III — 13 points

Write a main program that declares and reads an array of N integers, and then sets to 0 the duplicated appearance of all its elements.

Example:

If the entered array is: 1, 7, 1, 3, 5, 1, 7
The array should become 1, 7, 0, 3, 5, 0, 0

Note: the value 1 and the value 7 are duplicated, and replaced by 0 starting their second appearance.

Problem IV — 20 points

1. Write the function **isCapital** that takes a string and returns 1 if the string contains only (exclusively) capital alphabet, and 0 otherwise.
2. Write the function **Replace** that takes a string and replaces the most frequent capital letter by the least frequent capital letter in the string.

   Example: if the string is ABBCKBCDA, it should become AKKCKKCDA, as the most frequent letter is B and the least frequent is K.
   Note: if there is more than one (most frequent/least frequent), consider one on them.
3. Write a main function that keeps asking the user to enter a new string until the entered string contains only capital letters, then applies the replacement by the use of the function Replace, and prints the new string.

Good ***Luck***

LEBANESE UNIVERSITY

Faculty of Sciences – Section I

الجامعة اللبنانيّة

كلية العلوم - الفرع الأول

Final: ***I1101 En***

Duration: 2h

1st year – 2nd semester 2016

Date: 07/06/2016

Exercise I (25 pts)

Proper divisors of a positive integer N are all divisors of N other than N. For example: the proper divisors of 10 are 1, 2 and 5. An integer triplet (A, B, C) is said to be *friendly* if and only if the sum of proper divisors of one is equal to the sum of proper divisors of the other two. For example: The triplet (5, 8, 10) is a triplet of friendly integers, indeed:

N	Proper Divisor	Sum of Proper Divisors	
5	1	1	8=1+7
8	1, 2, 4	7	
10	1, 2, 5	8	

a) Write a function **friendly** that accepts as parameter three positive integers (A, B, C) and returns 1 if the three form a friendly integer triplet, 0 otherwise.

b) Write a main program that asks the user to enter three integers A, B and C and then checks if the triplet (A, B, C) is friendly.

Exercise II - Lab (20 pts)

a) Write a function **numberOfVowels** that takes as parameter a text T (maximum 50 characters), and returns the number of vowels in T.

b) Write a function **numberOfLetters** that takes as parameter a text T (maximum 50 characters), and returns the number of letters in T.

c) Write a function **percentage** that takes as parameter a text T (maximum 50 characters), uses the above functions to calculate and display the percentage of letters, vowels and consonants in T.

Example:
If the text is: "I love Code for Win!"
numberOfVowels returns 7
numberOfLettres returns 15
percentage displays: The percentage of letters in this text is: 75%
The percentage of vowels is this text is: 35%
The percentage of consonants in this text is: 40%

Exercise III (25 pts)

Let T be an integer array of maximum size N.

a) Write a function **withoutDuplicate** that returns 1 if T is without duplicates (i.e. no element appears more than one time in the array), 0 otherwise.

b) Write a function **internal** that returns 1 if $\forall i \in [0, n[,\ T[i] \in [0, n[$ and returns 0 otherwise. Where n is the number of integers in T.

c) T is said to be a *permutation* if $\forall i \in [0, n[, T[i] \in [0, n[$ and if T contains no duplicates. Write a function **permutation** that returns 1 if T is a permutation, 0 otherwise.

d) Write a C program that reads T and tests whether it is a permutation or not.

Example:
if T={1,4,3} T is not a permutation
if T={1,0,0} T is not a permutation
if T={1,0,2} T is a permutation

Good Luck

Programmation impérative

2éme semester 2016

Exercice 1:

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

a)

```
int Amiable(int a,int b,int c)
{
int sa=0,sb=0,sc=0,i;
  for(i=1;i<=a/2;i++)
     if(a%i==0)
        sa+=i;
 for(i=1;i<=b/2;i++)
     if(b%i==0)
        sb+=i;
 for(i=1;i<=c/2;i++)
     if(c%i==0)
        sc+=i;
if(sa==(sb+sc) || sb==(sa+sc) || sc==(sa+sb))
return 1;
return 0;
}
```

b)

```
void main()
{
int a,b,c;
```

```
printf("doner 3 entiers: ");
scanf("%d%d%d",&a,&b,&c);
if(Amiable(a,b,c)==1)
printf("(%d,%d,%d) forment un triplet amiable",a,b,c);
else
printf("(%d,%d,%d) ne forment pas un triplet amiable",a,b,c);
getch();}
```

Exercice 2:

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

a)

```
int nombreVoyelles(char t[51])
{
  int i,v=0;
    for(i=0;i<strlen(t);i++)
  if(t[i]=='a' || t[i]=='i' ||t[i]=='e'
||t[i]=='o'||t[i]=='u'||t[i]=='A'||t[i]=='I'||t[i]=='E'||t[i]=='O'||t[i]=='U')
    v++;
 return v;
}
```

b)

```
int nombreLettres(char t[51])
{
int i,l=0;
for(i=0;i<strlen(t);i++)
if(t[i]<'z' && t[i]>'A')
```

```
    l++;
    return l;}
```

c)

```
    void pourcentage(char t[51])
    {
    float pv,pl,pc;
    int c;
    pv=100.0*nombreVoyelles(t)/strlen(t);
    pl=100.0*nombreLettres(t)/strlen(t);
    c=nombreLettres(t)-nombreVoyelles(t);
    pc=100.0*c/strlen(t);
     printf("le pourcentage des lettres est %0.1f%%\nle pourcentage des voyelles est
    %0.1f%%\nle pourcentage
    des consonnes est %0.1f%%",pl,pv,pc);
    }
    void main()
        {
        char t[51];
     printf("donner le texte: ");
   gets(t);
   pourcentage(t);
   getch();
   }
```

Exercice 3:

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

a)

```
int sansDoublons(int T[100],int N)
{
int i,j,s=0;
for(i=0;i<N-1;i++)
for(j=i+1;j<N;j++)
if(T[i]==T[j])
s++;
if(s==0)
return 1;
return 0;
}
```

b)

```
int interne(int T[100],int N)
{
int i,s=0;
for(i=0;i<N;i++)
if(T[i]<0 || T[i]>N-1)
s++;
if(s==0)
return 1;
return 0;
}
```

c)

```
int permutation(int T[100],int N)
{
return sansDoublons(T,N)*interne(T,N);
```

```
}
```

d)

```
void main()
{
int T[100],N,i;
printf("donner la taille maximal du tableau: ");
scanf("%d",&N);
printf("donner les valeurs du tableau :");
for(i=0;i<N;i++)
scanf("%d",&T[i]);
if(permutation(T,N)==1) printf("T est une permutation");
else printf("T n est pas une permutation");
getch();
}
```