EXERCICI PREVI

Nota: *Feu aquest exercicis abans de la classe de laboratori.*

1. [Errors.cpp] Els programes següents són incorrectes o no donen el resultat esperat. Indiqueu per què i escriviu-los de manera correcta:

```
#include <iostream>
#include <stdlib.h>
using namespace std;

void es_parell(void)
{
    if((valor_llegit % 2) == 0)
        cout<< "El valor "<<valor_llegit<<" és parell\n";
    else
        cout<< "El valor "<<valor_llegit<<" és imparell\n";
}

int main(void)
{
    int valor_llegit;

    cout << "Introdueix un valor";
    cin >> valor_llegit;
    es_parell();

    system("PAUSE");
    return 0;
}
--------------------------------
#include <iostream>
#include <stdlib.h>
using namespace std;

void demana_caracter(char lletra)
{
    do
    {
        cout <<"Introdueix una lletra minúscula\n";
        cin >> lletra;
    }
    while((int(lletra)<int('a'))|| ((int(lletra)> int('z'));
}

int main(void)
{
    char meua_lletra  = '?';

    demana_caracter(meua_lletra);
    cout << "La lletra llegida és" <<meua_lletra<< endl;

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

EXERCICIS RESOLTS A CLASSE DE TEORIA

Nota: *En aquests exercicis cal que implementeu el pseudocodi o organigrama discutit en teoria.*

2. [MCM.cpp] **(Tema2-ExercicisFuncions-10)** Realitzeu un programa que calcule el mínim comú múltiple de dos nombres introduïts pel teclat. Useu la funció `mcd` implementada segons l'algorisme d'Euclides ja que cal recordar que:

$$MCM(a,b) = \frac{a*b}{mcd(a,b)}$$

3. [Combinatoris.cpp] **(Tema2-ExercicisFuncions-13)** Escriviu un programa que calcule el nombre combinatori de *m* sobre *n*. Dividiu el programa en funcions (empreu una funció que calcule el factorial d'un nombre que rep com a paràmetre).

$$C_m^n = C_{m,n} = C(m,n) = \binom{m}{n} = \frac{m!}{n!(m-n)!}$$

EXERCICIS PER RESOLDRE

Nota: *En aquests exercicis cal que dissenyeu el vostre propi pseudocodi o organigrama*

4. [EsPrimer.cpp] Escriviu una funció que retorne si un nombre, que se li passa com a paràmetre, és primer o no. Empreu-la per fer un programa que, donat un nombre introduït pel teclat, mostre per pantalla tots els nombres primers inferiors a aquest. (Adaptació a funcions de l'exercici 3 de la pràctica 4),

5. [Divisio.cpp] Escriviu una funció que, donats dos enters positius x i y, calcule la divisió entera i el residu de la divisió fent servir només l'operador de subtracció. (Adaptació a funcions de l'exercici 5 de la pràctica 4).

6. [PotenciaV2.cpp] Escriviu un programa que continga la funció `potencia(x,y)`, implementada per vosaltres, que calcule el valor de x elevat a y mitjançant multiplicacions, on x i y són dos nombres enters. Feu-la servir en el programa principal per mostrar x^y i y^x. No es pot emprar la funció `pow` de la llibreria matemàtica, excepte si voleu comprovar els resultats. (Adaptació de l'exercici 7 de la pràctica 4).

7. [Aleatoris.cpp] Si volem obtindre un nombre aleatori dins d'un rang determinat [R1…R2], podem cridar la funció `rand` i modificar el valor que torna de la manera següent:

$y = \frac{x \cdot (R2 - R1)}{RAND\_MAX} + R1$, on `x= rand()`, $x \in [0 \ldots RAND\_MAX]$, $y \in [R1 \ldots R2]$

Per iniciar el generador de nombres aleatoris sense haver de proporcionar explícitament una llavor diferent per a cada execució, podem cridar `srand(time(NULL))`, ja que `time(NULL)` torna l'hora actual en dècimes de segon. Feu una funció `aleat(n,m)` que genere enters aleatoris en el rang `[n…m]` i que torne el nombre aleatori generat. A continuació, feu un programa que empre aquesta funció per generar i mostrar per pantalla 10 nombres aleatoris en un rang triat per l'usuari.

Nota: Per a podar usar la funció *`time`* heu d'incloure la llibreria `time.h`

8. [Endevina.cpp] Escriviu un programa que genere un nombre aleatori entre 0 i 99 i que demane a l'usuari que l'endevine tot i dient quants intents li han fet falta. L'ordinador donarà pistes al jugador sobre si el nombre que ha introduït és major o menor que el que ha d'endevinar. Empreu la funció creada en l'exercici "Aleatoris.cpp". (Adaptació a funcions de l'exercici 12 de la pràctica 4).

EXERCICIS DE REFORÇ

Nota: *Recomanem que els feu per a aconseguir un coneixement més sòlid sobre l'assumpte*

9. [Nombree.cpp] **(Tema2-ExercicisFuncions-16)** El valor de e^x es pot aproximar mitjançant el sumatori següent:

$$e^x = \sum_{i=0}^{n} \frac{x^i}{i!}$$

Escriviu un programa que demane a l'usuari el valor de *x* i el valor de *n* i mostre per pantalla el valor de l'aproximació de e^x per a la *x* i la *n* introduïdes. La funció principal haurà de cridar a una funció `aproximacio(x,n)` que retorne el valor buscat. La funció `aproximacio(x,n)` ha d'emprar a la seua vegada les funcions `factorial(x)` i `potencia(x,i)` dels exercicis anteriors. Compareu el resultat de l'aproximació calculada amb el resultat que s'obté mitjançant la funció `exp(x)` de la llibreria matemàtica i mostreu aquesta diferència per pantalla.

10. [CercaAmics.cpp] Escriviu un programa que demane un nombre natural i cerque els seus nombres amics en un rang [min,max] que també demanarà a l'usuari. El programa mostrarà per pantalla únicament els nombres que són amics o indicarà si no n'ha trobat cap.

Recordatori: Dos nombres a i b són amics si la suma dels divisors de a (llevat ell mateix) coincideix amb b i viceversa.

Caldrà fer dos funcions: la funció *divisors* que retornarà la suma dels divisors d'un nombre i la funció *amics* que retornarà un valor booleà que indicarà si dos nombres són amics.

11. [Distancia.cpp] Escriviu un programa que calcule la distància de dos punts descrits per les seues coordenades (x_1,y_1) i (x_2,y_2). Haureu de crear una funció de l'estil `distancia(x1,y1,x2,y2)`, que serà cridada des de la funció principal i tornarà el valor de la distància. Empreu, a més a més, una funció per a llegir les coordenades d'un punt.

12. [Gravetat.cpp] Escriviu un programa que calcule la força d'atracció gravitatòria entre dues masses m1 i m2 situades en les coordenades (x1,y1) i (x2,y2). Caldrà crear una funció `fgrav(m1,x1,y1,m2,x2,y2)`, que serà invocada des del programa principal i tornarà el valor de la força. La funció `fgrav` cridarà a la funció `distància(x1,y1,x2,y2)` de l'exercici "Distancia.cpp".