EJEMPLOS PARA PRACTICAR EL EXAMEN PARCIAL

A continuación se presentan 9 problemas usados en examenes parciales anteriores. Cada examen parcial consta de tres ejercicios, uno del tipo I (3 puntos), otro del tipo II (4 puntos y otro del tipo III (3 puntos). La idea es orientar a los alum,nos en el tipo de examen y sobre todo **los conocimientos y habilidades** que se evalúan. No obstante, tomad en cuenta que el número y tipo de ejercicios podrían cambiar buscando una mejor evaluación y la originalidad requerida por la calidad formativa.

PROBLEMA TIPO I – A

El següent programa determina el número de dies que separen dues dates. L'entrada són dues dates en format DD/MM/AAAA, i se asume que la primera data és sempre anterior a la segona. El programa avança dia a dia des de la data inicial fins a la data final comptant els dies que transcorren utilitzat tres subprogrames: un que determina si un any és de traspàs, un altre que retorna el nombre de dies d'un mes i un altre per a incrementar una data. Un any és de traspàs si és divisible entre quatre i no és divisible entre 100 o és divisible entre 400.

Completa les expressions omeses, marcades amb **#N#**, per a que el programa funcioni correctament. **#N#** representa codi que falta, aquest codi pot ser un símbol (per exemple >=), una instrucció (per exemple **float** xx, pi=3.1416;) o part d'una instrucció (per exemple: **else if**(a==0). Mai representa més d'una instrucció.

```
#include <iostream>
using namespace std;

#1# traspas(int a);
int dies_mes(int m, int a);
#2#
int main() {
    #3#
    int dia, mes, any;
    int d_ini, m_ini, a_ini, d_fin, m_fin, a_fin, dies = 0;
    cin >> d_ini >> c >> m_ini >> c >> a_ini;
    #4#
    dia= d_ini; mes= m_ini; any= a_ini;
    while (#5#) {
        incr_dia(dia, mes, any);
        dies++;}
    cout << "Nombre de dies = " << dies;
    if (#6#) cout << " dies";
    else cout << " dia";
    cout << endl;
}

#1# traspas(int a) {
    return ((#7#) or a%400==0);}

int dies_mes(int m, int a) {
    int dies = 31;
    if (m == 4 || m == 6 || m == 9 || m == 11) dies = 30;
    else if (m == 2) {
         if (#8#) dies = 29;
         else dies= 28; }
    return dies;}

void incr_dia(#9#) {
    dia++;
    if (dia > #10#) {

        dia=1;
        mes++;
        if (mes>12) {
            mes = 1;
            any++;} } }
```

PROBLEMA TIPO II - A

a) Implementeu un subprograma que donades les coordenades (*x,y*) de *2* punts, retorni la distància entre ells. Es pot utilitzar la funció `sqrt()` (arrel quadrada) de la biblioteca `<cmath>`.
b) Implementeu un subprograma que donades les coordenades (x,y) de *3* punts que corresponen als *3* vèrtexs d'un triangle i utilitzant el subprograma implementat en a) determini si es tracta d'un triangle equilàter (tots els costats iguals), isòsceles (dos costats iguals) o escalè (tots els costats diferents). El subprograma actualitzarà llavors algun dels tres comptadors de tipus de triangle.
c) Escriure un programa que llegeixi una llista d'elements, on cada element està format per els tres punts (*x,y*) de un triangle en l'espai *2D*. La llista és llegirà del fitxer `"Triangles.txt"`. El programa utilitzarà el subprograma definit en el apartat b) per a tractar cada element i comptabilitzar el número de triangles equilàters, isòsceles o escalens. El programa haurà d'escriure:

    1. El percentatge d'elements que corresponen a cada tipus de triangle.
    2. La llargària del triangle equilàter més gran.

PROBLEMA TIPO III - A

Implementeu un subprograma en C++ que donats dos nombres enters positius *a* i *b*, retorni 3 valors:

- el nombre de divisors comuns de *a* i *b*.
- el major d'aquests divisors
- algun tipus de valor que digui si el major d'aquests divisors es un nombre primer, o no. Un nombre primer es un nombre enter superior a *1* que admet **exactament dos divisors**: *1* i ell mateix.

PROBLEMA TIPO I – B

El siguiente programa carga desde un fichero una secuencia de caracteres y muestra por pantalla si hay más letras mayúsculas que minúsculas, así como el porcentaje de vocales entre aquellos caracteres que sean una letra. Para ello utiliza un subprograma que, dada una letra, regresa si es mayúscula, minúscula y si es vocal o no. Lamentablemente el código está incompleto. Hay una marcas ## **n** ## que indican la falta de código i que necesitan ser remplazadas para que funcione correctamente. Puede ser un símbolo (por ejemplo, `>=`), una línea de código (por ejemplo, **`float xx, pi=3.1416;`**) o partes de una instrucción (por ejemplo: **`else if (a==0)`**). Nunca representan más de una instrucción. **Escriba la lista del código que falta**.

```
#include<iostream>
## 1 ##
using namespace std;

## 2 ##
int main()
{
    ifstream fin("caracters.txt");
    char ca;
    ## 3 ##

    int conta_let=0, conta_ma=0, conta_vo=0;
    while(## 4 ##)
    {
        if(ca>='A' && ca<='Z' ## 5 ##){
            ## 6 ##
            deletras(ca,esmima,esmivo);
            if(esmima==true) conta_ma++;
            if(esmivo==true) conta_vo++;
        }
    }

    cout << conta_let << endl << conta_vo << endl;
    if (conta_let==0) cout << "NO HI HA CAP LLETRA EN EL FITXER" << endl;
    else
    {
       cout << "EL PERC. DE VOCALS ES: " << ## 7 ## << endl;
       if (conta_ma >= ## 8 ##)
            cout << "HI HA MES (o MATEIXES) MAJUSCULES QUE MINUSCULES" << endl;
       else cout << "HI HA MENYS MAJUSCULES QUE MINUSCULES" << endl;
    }
}

void deletras(char le ## 9 ##)
{
    es_MAY=false;
    es_voc=false;
    if (## 10 ##)
    {
        es_MAY=true;
        if(le=='A' || le=='E' || le=='I' || le=='O' || le=='U') es_voc=true;
    }
    else
    {
```

PROBLEMA TIPO II – B

Imitaremos una calculadora usando la información del fichero **"`Operacions.txt`"**, en cada línea tenemos dos **operandos** (los números que se operan) y un carácter indicando un **operador** (la operación aritmética que es realiza). Se solicita:

a) *(1 punto)* desarrollar un programa que obtenga los resultado de cada una de las operaciones del fichero. Cada operación consta de tres entradas: **un operando** seguido por **un operador** de los seis permitidos (`+, -, *, /, %, ^`), seguido **de otro operando**, siendo los operandos números enteros. Algunos ejemplos de operaciones son: `3 - 5, -9 % 2 i -2 ^ 8`.

La dificultad es que para hacer el programa, o cualquier subprograma, **se exige que únicamente se usen las tres operaciones permitidas: la suma +, la resta – i el cambio de signo.** Solamente para la suma y la resta se pueden usar directamente los operadores de *C++* conseguidos, mientras que para el resto de las operaciones será necesario crear los subprogramas siguientes (donde no está permitido ni utilizar los operadores `*, / o %,` ni tampoco ninguna biblioteca matemática);

b) **`PRODucte`** *(0,75 puntos)* – recibe dos enteros y regresa el producto de estos.

c) **`POTencia`** *(0,75 puntos)* – recibe dos enteros: ***`base`*** y ***`exp`*** y regresa el resultado de la potencia $base^{exp}$. La única restricción es que ***`exp`*** sea mayor o igual a cero. Se puede usar el subprograma PRODucte.

d) **`DIVisio`** *(1,25 puntos)* – recibe dos enteros ***`D`*** (dividendo) y ***`d`*** (divisor) i regresa dos enteros como resultados: el cociente y su residuo. En cas que ***`d`*** sea igual a cero, el subprograma regresará las constantes (ya conocidas por el compilador de C++) **`INT_MAX`** o **`INT_MIN`** dependiendo del signo de ***`D`***, los cuales son los valores posibles máximo y mínimo del tipo **`int`** en C++. Por definición, el signo del residuo es el mismo signo del dividendo. Ejemplo se entradas del fichero **"`Operacions.txt`"** y sys salidas correspondientes son:

Entrada	**Sortida**	**Entrada**	**Sortida**	**Entrada**	**Sortida**
`8 + 6`	`Suma = 14`	`2 / 3`	`Quocient = 0`	`9 % -2`	`Residu = 1`
`-2 - 6`	`Resta = -8`	`2 % 3`	`Residu = 2`	`-9 / 0`	`Quocient = -2147483648`
`-2 - -6`	`Resta = 4`	`8 / 2`	`Quocient = 4`	`-9 % 0`	`Residu = 0`
`2 * 0`	`Producte = 0`	`8 % 2`	`Residu = 0`	`9 / 0`	`Quocient = 2147483647`
`0 * 3`	`Producte = 0`	`9 / 4`	`Quocient = 2`	`9 % 0`	`Residu = 0`
`2 * 0`	`Producte = 0`	`9 % 4`	`Residu = 1`	`2 @ 3`	`Operacio Invalida`
`0 * 0`	`Producte = 0`	`-9 / 2`	`Quocient = -4`	`2 ^ 0`	`Potencia = 1`
`2 * 3`	`Producte = 6`	`-9 % 2`	`Residu = -1`	`0 ^ 3`	`Potencia = 0`
`-2 * 3`	`Producte = -6`	`-9 / -2`	`Quocient = 4`	`2 ^ 3`	`Potencia = 8`
`2 * -3`	`Producte = -6`	`-9 % -2`	`Residu = -1`	`-2 ^ 3`	`Potencia = -8`
`-2 * -3`	`Producte = 6`	`9 / -2`	`Quocient = -4`	`0 ^ 0`	`Potencia = 1`
`0 / 3`	`Quocient = 0`	`3 / 3`	`Quocient = 0`		
`0 % 3`	`Residu = 0`	`3 % 3`	`Residuo = 0`		

PROBLEMA TIPO III – B

Un número entero $n > 0$ es **abundante** si es menor que la suma de sus divisores propis (i.e. todos sus divisores excepto el propio n); se dice en cambio que n es **deficiente** si es mayor que la suma sus divisores propios. Si n no es ni abundante ni deficiente, entonces es **perfecto**. Per ejemple:

- *12* es abundante, porque *1,2,3,4,6* son sus divisores propios y $1+2+3+4+6=16 > 12$.
- *15* es deficiente porque *1,3,5* son sus divisores propios y $1+3+5=9 < 15$.
- *6* es perfecto porque *1,2,3* son sus divisores propios y $1+2+3=6 == 6$.

Escribir un subprograma (acción o función) que, dados dos valores enteros positivos a i b con $a < b$, regrese tres valores: la cantidad de números abundantes, la cantidad de números deficientes y la cantidad de números perfectos que hay en el intervalo $[a,b]$.

Nota histórica: Los números naturales van ser clasificados en deficientes, abundantes i perfectos per *Nicomacus* en el su *Introductio Arithmetica* (circa 100 AD).

PROBLEMA TIPO I – C

El siguiente programa resuelve este problema: se introduce por teclado un número entero *N* seguido de *N* números enteros mayores que 1, para los cuales se calcula el mínimo común múltiplo de todos ellos. No obstante, el programa está incompleto. Hay unas marcas **#n#** que indican que falta código y que se ha de completar para que funcione correctamente. El código faltante puede ser un símbolo (por ejemplo >=), una línea (por ejemplo **float** xx, pi=*3.1416*;) o partes de una instrucción (por ejemplo: **else if** (a==0)). En ningún caso representa más de una instrucción. Escriba la lista de códigos que faltan.

```
#1#
using namespace std;

#2#
int main(){
  int N, mcm, #3#;
  cout<< "Para cuantos enteros > 1 se calculara el mcm?"<<endl;
  cin>> N;
  cout<< "Introduzca los "<<N<<" numeros enteros"<<endl;
  mcm = #4#;
  for(int k=0; #5#;k++){
     #6#;
     mcm = MCM(num,mcm);
  }
  cout<<endl<<"El minimo comun multiplo es "<< mcm<<endl;
}

int MCM(int a, #7#){
  int multpl #8#;
  while(multpl <= a){
    if (b*multpl%a == 0){return #9#;}
    #10#;
  }
}
```

PROBLEMA TIPO II – C

Es demana desenvolupar un programa (**1,5 punts**) que classifiqui un nombre indeterminat de punts 2D, en les categories **A, B** o **C** definides com (veure figura):

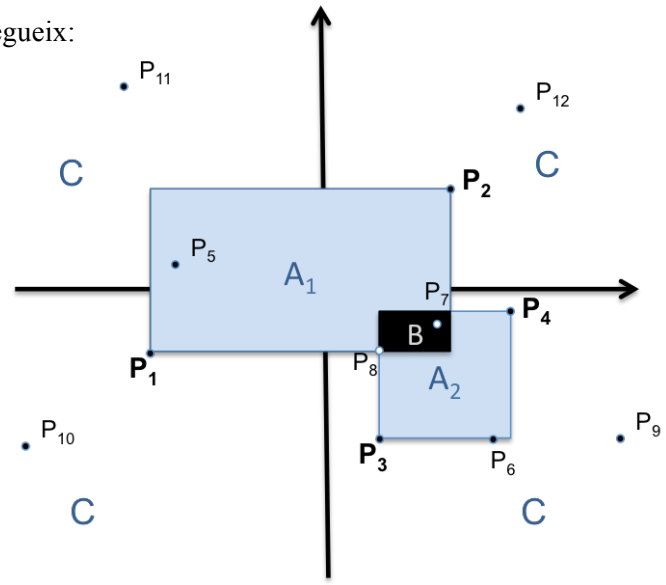
- **A.** aquesta àrea és la unió de les àrees que ocupen els rectangles $\mathbf{A}_1$ y $\mathbf{A}_2$ incloent els seus perímetres però exclou l'àrea de la seva possible intersecció
- **B.** aquesta àrea (que pot ser nul·la) és la definida per la intersecció dels rectangles $\mathbf{A}_1$ y $\mathbf{A}_2$ que **inclou el perímetre de la seva intersecció**.
- **C.** aquesta àrea és la definida com l'àrea fora de les àrees ocupades pels rectangles $\mathbf{A}_1$ y $\mathbf{A}_2$.

Totes les dades seran llegides des del fitxer `"Data.txt"` el qual tindrà inicialment, i en ordre, els parells de coordenades (*x, y*) dels punts $\mathbf{P}_1$ i $\mathbf{P}_2$ que defineixen el vèrtex del rectagle $\mathbf{A}_1$, seguides dels parells de coordenades (*x, y*) dels punts $\mathbf{P}_3$ i $\mathbf{P}_4$ que defineixen el vèrtex del rectangle $\mathbf{A}_2$ seguits d'un nombre indeterminat de parells (*x, y*) amb les coordenades dels punts a classificar. Cal tenir en compte que els rectangles poden, o no, fer intersecció. El programa ha de mostrar per pantalla el percentatge de punts que es troben en cada categoria i ha de cobrir qualsevol possibilitat.

Per a l'anterior, es requereix dissenyar i utilitzar un subprograma (**1,5 punts**) que, donades les coordenades (*x, y*) del rectangles $\mathbf{A}_1$ y $\mathbf{A}_2$ i les coordenades (*x, y*) d'un punt, ens retorni tres valors que diguin respectivament si el punt està o no, dins de les categories **A, B** o **C**. S'ha de tenir en compte que cap punt pot estar en més d'una categoria.

En l'exemple els punts del $\mathbf{P_5}$ al $\mathbf{P_{12}}$ classifiquen com segueix:

- **A.** $\mathbf{P_5}$, $\mathbf{P_6}$
- **B.** $\mathbf{P_7}$, $\mathbf{P_8}$
- **C.** $\mathbf{P_9}$, $\mathbf{P_{10}}$, $\mathbf{P_{11}}$, $\mathbf{P_{12}}$



Resultats:

```
Grup A: 25%
Grup B: 25%
Grup C: 50%
```

PROBLEMA TIPO III – C

Un nombre enter $p > 1$ és **primer** si només és divisible per *1* i per el mateix. Escriure un subprograma (acció o funció) que, donats dos valors sencers positius *a* i *b* amb $a < b$, retorni dos valors: la quantitat de nombres primers *p* tal que *p+2* es també primer (com *3 i 5, 5 i 7, o 11 i 13*) i la quantitat de nombres primers *p* tal que *p+2* **no** es primer que hi ha en l'interval $[a,b]$.

Ajuda: pot ser convenient escriure una funció que determini si un número és primer o no.