FUNDAMENTOS DE PROGRAMACIÓN
Convocatoria I – curso 19/20
Examen Teórico – **Modelo A**
Duración: 2.5 horas
No se puede desgrapar el examen

Puntuación:	Nombre y Apellidos del alumno: DNI:

1. **(1 punto)** Realiza el seguimiento del código mostrado a continuación e indica qué resultado mostrará por pantalla tras su ejecución. **Se debe contestar en el recuadro del enunciado**.

```
#include <iostream>
using namespace std;
class Point
{ int A, B, C;
public:
  Point() {A=B=C=0;}
  Point(int a) {A=a; B=C=0;}
  Point(int a, int b) {A=a; B=b; C=0;}
  Point(int a, int b, int c) {A=a; B=b; C=c;}
  void Get (int &a, int &b, int &c) {a=A; b=B; c=C;}
  void MathOper (int a, int b, int c) {A+=a; B+=b; C+=c;}
  Point MathOper (Point p) {p.MathOper(A,B,C); return p;}
  void Show() {cout << "A=" << A << " B=" << B << " C=" << C << endl;}
};
int main()
{ Point P1, P2(2,5), P3(3,4,5);
  P2.Show();
  P3.Show();
  P1 = P2.MathOper(P3);
  P1.Show();
  return 0;
}
```

Solución:

2. **(4 puntos)** Necesitamos implementar una clase llamada **`Tabla`** con las siguientes especificaciones:

- **Atributos privados**:
  - **`T`**, que será una tabla bidimensional de **`5x10`** elementos de tipo **`entero`**.

- **Métodos públicos**:
  - **`Constructor`**.
    Con un parámetro de entrada **`v`** de tipo **`entero`**, deberá **`rellenar`** el atributo **`T`** con **`valores aleatorios`** comprendidos entre **`1`** y **`v`**.
  - **`Mostrar2D.`**
    Sin parámetros, deberá **`mostrar por pantalla`** en formato **`2D`** el atributo **`T`**.
  - **`MaximoF.`**
    Con un parámetro de entrada **`f`** y un parámetro de E/S **`c`**, ambos de tipo **`entero`**, deberá **`devolver`** el valor **`máximo`** de la fila **`f`** y a través del parámetro **`c`** **`devolverá`** en qué **`columna`** se encuentra dicho valor.
  - **`MinimoC.`**
    Con un parámetro de entrada **`c`** y uno de E/S **`mc`**, ambos **`enteros`**, deberá **`devolver`** a través del parámetro **`mc`** el valor **`mínimo`** de la columna indicada por el parámetro **`c`**.

Una vez implementada la clase **`Tabla`**, realizamos una función **`main()`** que haciendo uso de dicha clase debería ser capaz de indicar si existe o no un **Punto de Silla**, en el atributo **`T`**. En el caso de existir Punto de Silla, nuestro programa sólo tiene que encontrar el primero y mostrar su posición **`(fila, columna)`**. Si no existiera ninguno se le comunicaría al usuario con un mensaje.

Se ha trabajado mucho en la implementación tanto de la clase como de la función **`main()`**, sin embargo, pese a nuestros esfuerzos el programa no funciona. Por este motivo, **se pide solucionar no sólo los errores sintácticos sino también de implementación** que existen **en todo el código** proporcionado. **La solución se debe realizar en el enunciado según se indica en el recuadro**.

Se le recuerda al alumno que un Punto de Silla es aquel elemento **`[i][j]`** cuyo valor es el máximo de la fila **`i`** y el mínimo de la columna **`j`**.

Un ejemplo de ejecución correcta del programa sería la siguiente:

```
15  16  17  18  19  20  21  22  23  24
25  17  18  19  20  21  22  23  24  25
26  18  19  20  21  22  23  24  25  26
27  19  20  21  22  23  24  25  26  27
28  20  21  22  23  24  25  26  27  28

Existe un Punto de Silla en la posicion (0,9)
```

Máximo de su fila y mínimo de su columna

Importante:

El alumno deberá tachar la línea incorrecta y escribirla correctamente a su lado. En el caso de que deba incluir alguna línea de código nueva y no hubiese hueco se procederá del siguiente modo:

Supongamos que falta **`#include <cstdlib>`** y que debería ir entre la línea 1 y la 2

```
1    #include <iostream>
1-2 ------------------------> #include <cstdlib>
2    using namespace std;
```

```
1    #include <iostream>
2    #include <cstdlib>
3    #include <ctime>
4    using namespace std;
5
6    #define NumF 5;
7    #define NumC 10;
8
9    class Tabla
10   {
11      int T[NumF][NumC];
12   public:
13      void Tabla(int v);
14      void Mostrar2D();
15      int MaximoF (int f, int c);
16      void MinimoC (int c, int mc);
17   };
18
19   void Tabla(int v)
20   {
21     srand(time(NULL));
22     for (int i=NumF; i>=0; i--)
23        for (int j=NumC; j>=0; j--)
24           T[i][j] = rand() % v + 1;
25   }
26
27   void Mostrar2D()
28   {
29     for (int i=0; i<NumF; i+1)
30        for (int j=0; j<NumC; j+1)
31            cout << T[i][j] << " ";
32   }
33
```

```
34  int MaximoF (int f, int c)
35  {
36     int M, j;
37
38     M=T[f][0];
39     c=0;
40     j=1;
41     while (j<NumC);
42     {
43       if (T[f][j]>M)
44         M=T[f][j];
45         c=j;
46       else
47         j++;
48     }
49  }
50
51  void MinimoC (int c, int mc)
52  {
53     mc=T[0][c];
54     for (int i=1; i<NumF; i++)
55       if (T[i][c] < mc)
56          mc = T[i][c];
57  }
```

```
58
59  int main()
60  {
61     Tabla Datos(33);
62     int Fila, cMax, Maximo, Minimo;
63
64     Datos.Mostrar2D();
65     Fila =0;
66     HayPtoSilla = False;
67     while (Fila<NumF && !HayPtoSilla)
68     {
69        Maximo =  Datos.MaximoF(Fila, &cMax);
70        if (Maximo == Datos.MinimoC(cMax, Minimo))
71          HayPtoSilla = True;
72        else
73          Fila++;
74     }
75
76     if (HayPtoSilla)
77        cout << "Existe un Punto de Silla en la posicion (" << Fila
78             << "," << cMax << ")";
79     else
80        cout << "No existe ningun Punto de Silla";
81
82  return 0;
83  }
```

3. **(5 puntos)** Se desea recoger la información referente a las inspecciones realizadas en una determinada estación de revisiones, la información de los vehículos, así como de las inspecciones realizadas a cada uno. Para ello se utilizarán las estructuras que se detallan a continuación:

```
typedef char cadena[30];
struct fecha{
    int dia;
    int mes;
    int anyo;
};
struct vehiculo{
    cadena matricula; // Matrícula del vehículo
    int anyoFabricacion; // Año de fabricación
    bool esFurgoneta; // Verdadero si es furgoneta o por contra sería un coche
};
struct inspeccion{
    vehiculo v; // Vehículo sobre el que se realiza la inspección
    fecha fechaInspeccion; // Fecha en la que se realiza la inspección
    bool revFavorable; // Resultado en la revisión, verdadero favorable o por contra desfavorable
    int nRevision; // Número de revisión: 1 ó 2 (si un vehículo no pasa la revisión en la primera
                   // se le realizaría una segunda revisión)
};
```

La siguiente clase, controla y gestiona las inspecciones realizadas en una estación determinada:

```
class Estacion{
    cadena nombre; // Nombre de la estación
    inspeccion inspecciones[10000]; // Información de cada inspección realizada
    int nInspecciones; // Número de inspecciones realizadas
public:

    /* Evalúa la inspección situada en la posición pos, pasada como parámetro, para devolver el
    número de meses de vigencia que se estableció tras su revisión. El parámetro pos se supone
    que recibe un valor correcto. Si la revisión no fue favorable, devolvería el valor -1.
    Si fue favorable, el número de meses devuelto, en caso de ser un coche, sería 12 ó 24, en
    función de los años transcurridos entre el año de inspección (no tener en cuenta mes y día)
    y el año de fabricación del vehículo. Si dicho tiempo es superior estrictamente a 4 años,
    serían 12 meses, en caso contrario 24. Si el vehículo fuese una furgoneta, el tiempo anterior
    sería la mitad, 6 ó 12 */
    int vigenciaInspeccion(int pos);

    /* Devuelve en el parámetro sr, el porcentaje de segundas revisiones a furgonetas
    cuyo resultado de revisión haya sido desfavorable (porcentaje de rechazo)
    Si no hubiera ninguna furgoneta en 2ª revisión, el valor devuelto debería ser -1*/
    void porcentajeRechazoFurgonetas(float &sr);

    /* Muestra por pantalla, la fecha completa de la inspección en formato dd/mm/yyyy, así como
    el resultado obtenido, con el texto Favorable o No Favorable, en las n primeras inspecciones
    del vehículo cuya matrícula corresponda a la pasada en el parámtero mat.
    Ejemplo para cada salida de inspección:
         Fecha Inspec.: 11/5/2004 Resultado Inspec.: No Favorable
    Si hay menos de n inspecciones, mostrar además un mensaje de texto que indique el número de
    inspecciones encontradas. Si no existieran inspecciones para dicha matrícula, indicarlo
    con el mensaje de salida "No existen inspecciones para la matrícula indicada" */
    void listarInspecciones(cadena mat, int n);
};
```

Se pide **implementar** los métodos **vigenciaInspeccion (1 punto)**, **porcentajeRechazoFurgonetas(2 puntos)** y **listarInspecciones(2 puntos)**.

SOLUCIONES A LOS EJERCICIOS

FUNDAMENTOS DE PROGRAMACIÓN
Convocatoria I – curso 19/20
Examen Teórico – Modelo A
Duración: 2.5 horas
No se puede desgrapar el examen

Puntuación:	Nombre y Apellidos del alumno: **SOLUCIÓN** DNI:

1. **(1 punto)** Realiza el seguimiento del código mostrado a continuación e indica qué resultado mostrará por pantalla tras su ejecución. Se debe contestar en el recuadro del enunciado.

```
#include <iostream>
using namespace std;
class Point
{ int A, B, C;
public:
  Point() {A=B=C=0;}
  Point(int a) {A=a; B=C=0;}
  Point(int a, int b) {A=a; B=b; C=0;}
  Point(int a, int b, int c) {A=a; B=b; C=c;}
  void Get (int &a, int &b, int &c) {a=A; b=B; c=C;}
  void MathOper (int a, int b, int c) {A+=a; B+=b; C+=c;}
  Point MathOper (Point p) {p.MathOper(A,B,C); return p;}
  void Show() {cout << "A=" << A << " B=" << B << " C=" << C << endl;}
};
int main()
{ Point P1, P2(2,5), P3(3,4,5);
  P2.Show();
  P3.Show();
  P1 = P2.MathOper(P3);
  P1.Show();
  return 0;
}
```

Solución Modelo A:

A=2 B=5 C=0
A=3 B=4 C=5
A=5 B=9 C=5

```
1    #include <iostream>
2    #include <cstdlib>
3    #include <ctime>
4    using namespace std;
5    #define NumF 5 ;    (quitar ;)
6    #define NumC 10;    (quitar ;)
7
8    class Tabla
9    {
10      int T[NumF][NumC];
11   public:
12      void Tabla(int v);  (quitar void)
13      void Mostrar2D();
14      int MaximoF (int f, int &c);  (&)
15      void MinimoC (int c, int &mc); (&)
16   };
17
18   void Tabla::Tabla(int v)  (quitar void) y  (Tabla::)
19   {
20     srand(time(NULL));
21     for (int i=NumF-1; i>=0; i--)     (poner → -1)
22        for (int j=NumC-1; j>=0; j--)  (poner → -1)
23             T[i][j] = rand() % v + 1;
24   }
25
26   void Tabla::Mostrar2D()  (Tabla::)
27   {
28     for (int i=0; i<NumF; i++1 i++)        (i++)
29     {    for (int j=0; j<NumC; j++1 j++)    (j++)   y  (poner → { )
30             cout << T[i][j] << " ";
31          cout << endl;              (poner → cout << endl;)
32     }                                (poner →  } )
33   }
34
35   int Tabla::MaximoF (int f, int &c)   (Tabla::)  y (&)
36   {
37      int M, j;
38      M=T[f][0];
39      c=0;
40      j=1;
41      while (j<NumC);        (quitar ;)
42      {
43        if (T[f][j] > M)
44        {  M=T[f][j];   (poner → { )
45           c=j;
46        }                (poner → } )
47        else           (sobra else)
48          j++;
49      }
50      return M;    (poner → return M;)
51   }
52
```

```
53  void Tabla::MinimoC (int c, int &mc)  (Tabla::) y (&)
54  {
55     mc=T[0][c];
56     for (int i=1; i<NumF; i++)
57       if (T[i][c] < mc)
58         mc=T[i][c];
59  }
60
61  int main()
62  {  Tabla Datos(33);
63     int Fila, cMax, Maximo, Minimo;
64     bool HayPtoSilla;   (declarar HayPtoSilla de tipo bool)
65
66     Datos.Mostrar2D();
67     Fila =0;
68     HayPtoSilla = False;   (poner → false en minúscula)
69     while (Fila<NumF && !HayPtoSilla)
70     {
71         Maximo =  Datos.MaximoF(Fila, &cMax);   (quitar &)
72         Datos.MinimoC(cMax, Minimo)   (poner → la llamada)
73        if (Maximo == Datos.MinimoC(cMax, Minimo) Minimo) (usar)
74          HayPtoSilla = True;   (poner → en minúscula)
75        else
76          Fila++;
77     }
78     if (HayPtoSilla)
79        cout << "Existe un Punto de Silla en la posicion (" << Fila
80             << "," << cMax << ")";
81     else
82        cout << "No existe ningún Punto de Silla";
83
84  return 0;
85  }
```

/* Evalúa la inspección situada en la posición pos, pasada como parámetro, para devolver el

número de meses de vigencia que se estableció tras su revisión. El parámetro pos se supone

que recibe un valor correcto. Si la revisión no fue favorable, devolvería el valor -1.

Si fue favorable, el número de meses devuelto, en caso de ser un coche, sería 12 ó 24, en

función de los años transcurridos entre el año de inspección (no tener en cuenta mes y día)

y el año de fabricación del vehículo. Si dicho tiempo es superior estrictamente a 4 años,

serían 12 meses, en caso contrario 24. Si el vehículo fuese una furgoneta, el tiempo anterior

sería la mitad, 6 ó 12 */

int vigenciaInspeccion(int pos);

```
int Estacion::vigenciaInspeccion(int pos){
   int resp=-1;
   inspeccion inspec;
   inspec=inspecciones[pos];

   // Comprobamos si la ha pasado favorablemente
   if (inspec.revFavorable)
   {
      // Determinamos el tiempo que ha transcurrido entre la fecha de la revision
      // y la fecha de fabricación del vehículo
      if ((inspec.fechaInspeccion.anyo - inspec.v.anyoFabricacion)<=4)
         resp = 24;
      else
         resp = 12;

      // Si es una furgoneta, debería reducirse a la mitad el tiempo establecido
      // para un vehículo normal
      if (inspec.v.esFurgoneta)
         resp = resp / 2;
   }

   return resp;
};
```

Método 1

/* Devuelve en el parámetro sr, el porcentaje de segundas revisiones a furgonetas

cuyo resultado de revisión haya sido desfavorable (porcentaje de rechazo)

Si no hubiera ninguna furgoneta en 2ª revisión, el valor devuelto debería ser -1*/

void porcentajeRechazoFurgonetas(float &sr);

```
void Estacion::porcentajeRechazoFurgonetas(float &sr){
   int total=0, noFavorables=0;
   inspeccion inspec;

   for(int i=0; i< nInspecciones ; i++)
   {
      inspec = inspecciones[i];
      if (inspec.nRevision==2 && inspec.v.esFurgoneta){
         total++;
         if (!inspec.revFavorable)
            noFavorables++;
      }
   }

   if (total>0)
      sr= ((float)noFavorables / total) * 100;
   else
      sr=-1;
};
```

Método 2

/* Muestra por pantalla, la fecha completa de la inspección en formato dd/mm/yyyy, así como

el resultado obtenido, con el texto Favorable o No Favorable, en las n primeras inspecciones

del vehículo cuya matrícula corresponda a la pasada en el parámtero mat.

Ejemplo para cada salida de inspección:

Fecha Inspec.: 11/5/2004 Resultado Inspec.: No Favorable

Si hay menos de n inspecciones, mostrar además un mensaje de texto que indique el número de

inspecciones encontradas. Si no existieran inspecciones para dicha matrícula, indicarlo

con el mensaje de salida "No existen inspecciones para la matrícula indicada" */

void listarInspecciones(cadena mat, int n);

```
void Estacion::listarInspecciones(cadena mat, int n){
   int cont=0, i=0;
   inspeccion inspec;

   while ( (i<nInspecciones) && (cont<n) )
   {
      inspec=inspecciones[i];
      if (strcmp(mat, inspec.v.matricula)==0)
      {
         cont++;
         cout << "Fecha Inspec.: " << inspec.fechaInspeccion.dia << "/" <<
inspec.fechaInspeccion.mes
            << "/" << inspec.fechaInspeccion.anyo  << " ";

         cout << "Resultado Inspec.: ";
         if (inspec.revFavorable)
            cout << "Favorable";
         else
            cout << "No Favorable";
         cout << endl;

      }
      i++;
   }

   if (cont==0)
      cout << "No hay revisiones para la matricula dada." << endl;
   else if (cont<n)
      cout << "Se han encontrado tan solo " << cont << " revisiones." << endl;
};
```

Método 3