LENGUAJES DE PROGRAMACIÓN

INSTRUCCIONES
Por favor, entregue esta primera hoja de enunciado junto con el examen.
Dispone de 2 horas para realizar el examen.
MATERIAL PERMITIDO: Ninguno.

Pregunta 1 (3 puntos)

Indique la veracidad o falsedad de cada una de las afirmaciones siguientes, **explicando detalladamente** en cada caso el motivo de su respuesta.

A. (0.5 puntos) El control de flujo de Pascal se basa en el uso de las sentencias de salto (sentencias *goto*).

B. (0.5 puntos) Dado el tipo `color`, declarado en C++ de la siguiente manera:

```
enum color {blanco, azul, rojo};
```

El tipo `color` es un tipo enumerado que contiene las constantes literales blanco, azul y rojo cuyo valor es 0, 1 y 2, respectivamente.

C. (0.5 puntos) En el lenguaje Java el programador ha de indicar cuándo se debe eliminar una variable dinámica.

D. (0.5 puntos) Los lenguajes C++ y Python permiten al programador mantener el control de la ejecución del programa cuando se produce una excepción.

E. (0.5 puntos) La función de C++ `std::cin.peek()` devuelve el primer carácter del flujo de entrada, pero sin extraerlo del flujo.

F. (0.5 puntos) El algoritmo de ordenación por mezcla aplica el paradigma *divide y vencerás*.

Pregunta 2 (1 punto)

Escriba la salida por consola producida al ejecutar el programa en C++ escrito a continuación.

```
#include <iostream>
#include <vector>

int main() {
   int i = -1, *p1;
   std::vector<int> v(6, 2);
   std::vector<int>::iterator iter;
   p1 = &i;
   iter = v.begin() + 2;
   *iter = *p1;
   v.at(0) = 10;
   i = 8;
   *(iter + 2) = *p1;
   *p1 = -20;
   v.at(1) = i;
   std::cout << *iter << ", " << *(iter + 1)
             << ", " << *(iter + 2) << std::endl;
   std::cout << "v = ( " << v.at(0) << ", "
             << v.at(1) << ", " << v.at(2) << ", "
             << v.at(3) << ", " << v.at(4) << ", "
             << v.at(5) << ")" << std::endl;
   return 0;
}
```

Pregunta 3 (1 punto)

Escriba la salida por consola producida al ejecutar el programa en C++ escrito a continuación.

```
#include <iostream>

void fun1(int *a, int *b) {
   if (*a < *b) {
      int temp = *a;
      *a = *b;
      *b = temp;
   }
}

void fun2(int A[]) {
   std::cout << "A[0] = "  << A[0];
   std::cout << " A[1] = " << A[1] << std::endl;
   A[0] = -1;
   A[1] = -2;
}

int main() {
   int a[] = { 1, 2, 3 };
   int *p1, *p2;
   int i = 100, j = 200;
   p1 = &i;
   p2 = &j;
   fun1(p1, p2);
   std::cout << i << " " << j << std::endl;
   fun2(a);
   std::cout << "A[0] = " << a[0];
   std::cout << ", A[1] = " << a[1];
   std::cout << ", A[2] = " << a[2] << std::endl;
   fun2(&a[1]);
   std::cout << "A[0] = " << a[0];
   std::cout << "; A[1] = " << a[1];
   std::cout << "; A[2] = " << a[2] << std::endl;
   return 0;
}
```

Pregunta 4 (2.5 puntos)

Escriba un programa en C++ que gestione la entrega de billetes en el cajero automático de un banco. El programa debe realizar las acciones siguientes:

1. Abrir para lectura el fichero de texto *fondos.txt*, que almacena el número de billetes de cada valor que contiene en ese instante el cajero automático. En la primera columna del fichero está descrito el tipo de billete empleando las palabras *cinco*, *diez*, *veinte* y *cincuenta*, que describen el valor en euros del billete. En la segunda columna está el número total de billetes de cada tipo. Puesto que hay cuatro posibles tipos de billete (de valor 5, 10, 20 y 50 euros), el fichero tiene cuatro líneas. Si se produce error al abrir el fichero, mostrar un mensaje en la consola indicándolo y terminar.

2. Leer el fichero, almacenando su contenido en un mapa llamado `mFondos`. La clave es un string que indica el tipo de billete y el valor es un entero que indica el número de billetes de ese tipo que hay en el cajero automático. Cerrar el fichero.

3. Escribir un mensaje en la consola solicitando al cliente que introduzca la cantidad (en euros) que desea retirar del cajero. Leer de consola la cantidad introducida y almacenarla en una variable de tipo entero llamada `retirada`. Si la cantidad introducida no es múltiplo de 5, mostrar un mensaje de error y solicitar al cliente que introduzca nuevamente una cantidad. Esta operación puede repetirse tres veces. Si la tercera lectura es fallida, mostrar un mensaje indicándolo y terminar.

4. Calcular el mínimo número de billetes de distinto valor (50, 20, 10 y 5) necesario para entregar la cantidad solicitada, para lo cual deberá tenerse en cuenta el número de billetes de cada tipo disponibles en el cajero. Si no hay suficientes fondos en el cajero para completar la cantidad solicitada, el programa debe escribir un mensaje en la consola indicándolo y terminar.

5. Escribir en la consola el número de billetes de cada tipo dispuesto para satisfacer la cantidad solicitada.

6. Reescribir el fichero de texto *fondos.txt*, de modo que almacene el número de billetes de cada valor que contiene el cajero automático una vez realizada la entrega de billetes al cliente.

7. Terminar.

Pregunta 5 (2.5 puntos)

Escriba el código C++ indicado a continuación.

5.1 (0.5 puntos) Una función con el prototipo

```
CoordPunto3D fun1(double x, double y, double z);
```

donde el tipo estructura `CoordPunto3D`, definido de la forma siguiente

```
struct CoordPunto3D {
   double x, y, z;
};
```

almacena las coordenadas cartesianas de un punto en el espacio. La función declara una variable del tipo estructura `CoordPunto3D` llamada `coord`, asignándole a sus miembros las coordenadas cartesianas `x`, `y`, `z` pasadas como argumento a la función. La función devuelve la variable `coord`.

5.2 (0.5 puntos) Una función con el prototipo

```
int funcion1(int n, std::vector<int> &v)
          throw (std::invalid_argument);
```

que primeramente comprueba si el vector `v` tiene tamaño cero, en cuyo caso lanza una excepción. En caso contrario calcula, empleando el algoritmo *count*, y guarda, en una variable llamada `nElem`, el número de elementos del vector `v` cuyo valor es igual al valor del primer argumento de la función (`n`). Seguidamente, modifica el vector pasado por referencia (`v`), asignando el valor de `n` a todos los elementos del vector `v` cuyo valor sea mayor que `n`. Finalmente, devuelve el valor de la variable `nElem`.

5.3 (0.5 puntos) Una función con el prototipo

```
int funcion2(int n, std::vector<int> &a);
```

que ordene crecientemente los elementos del vector argumento pasado por referencia, y que calcule y devuelva el número de elementos del vector cuyo valor es mayor que el primer argumento de la función (`n`). Debe emplear el algoritmo *count_if* y una expresión lambda para realizar este último cálculo. Debe emplear el algoritmo *sort* para ordenar los elementos del vector.

5.4 (0.5 puntos) Una función con el prototipo

```
int funcion3(std::vector<CoordPunto3D> &v);
```

donde el tipo estructura `CoordPunto3D`, definido como se indicó en el Apartado 5.1, almacena las coordenadas cartesianas de un punto en el espacio. La función devuelve el número total de elementos del vector `v` que describen puntos cuya distancia al origen de coordenadas es menor que uno. Debe realizar el cálculo empleando el algoritmo count_if y una expresión lambda.

5.5 (0.5 puntos) Una función con el prototipo

```
std::list<double> funcion4(std::list<CoordPunto3D> &L);
```

donde el tipo estructura `CoordPunto3D` tiene la definición y el significado indicados en el Apartado 5.1. Cada elemento de la lista `L` representa un punto en el espacio, definido mediante sus coordenadas cartesianas. La función debe calcular la distancia al origen de coordenadas de cada elemento de la lista `L`, almacenar las distancias en una lista de elementos de tipo **double** llamada `lDist` y devolver la lista `lDist`. Debe realizar el cálculo empleando el algoritmo *transform* y una función lambda.