Nombre y Apellido: ..... N° Legajo: .....

Recuperatorio Primer Parcial de Programación Imperativa
25/11/2024

- **Condición mínima de aprobación: Sumar 5 (cinco) puntos.**
- **Los ejercicios que no se ajusten estrictamente al enunciado, no serán aceptados.**
- **No usar variables globales ni static.**
- **No es necesario escribir los #include**
- **Escribir en esta hoja Nombre, Apellido y Legajo**

Ejercicio 1 (3 puntos)

Implementar la función `void diffScores` que recibe un vector de `unsigned int` y su dimensión. Cada elemento del vector representa el puntaje obtenido por un jugador en una competencia. Se desea saber **cuál es la diferencia entre el puntaje más bajo** (el menor valor del vector) **y el puntaje más alto** (el mayor valor del vector), **expresada en centenas, decenas y unidades**.

Por ejemplo, si la diferencia fue de 731 puntos, el resultado debe ser 7 centenas, 3 decenas y 1 unidad.

La función debe **dejar en tres parámetros enteros de salida**:

- las centenas
- las decenas
- las unidades

El vector no está ordenado. Si el vector es vacío o tiene un único elemento, la diferencia entre el puntaje más bajo y el más alto es de 0 centenas, 0 decenas y 0 unidades.

Ejemplos:

Si el vector fuera `{150, 291, 853, 110, 852}`, el puntaje más bajo es `110` y el más alto es `853`, por lo que la diferencia es de 743 puntos (7 centenas, 4 decenas y 3 unidades).

Si el vector fuera `{150, 291, 1850, 110, 185}`, el puntaje más bajo es `110` y el más alto es `1850`, por lo que la diferencia es de 1740 puntos (17 centenas, 4 decenas y 0 unidades).

Ejercicio 2 (3,5 puntos)

Implementar la función `void` **`actualizarTareas`** que recibe dos vectores de enteros mayores a cero, ordenados en forma ascendente y sin repetidos, con un -1 como marca de final. El primer vector representa los códigos de las tareas a realizar y el segundo representa los códigos de las tareas que ya han sido realizadas.

La función debe **eliminar del primer vector las tareas que ya han sido realizadas**. Para verificar que el segundo vector sólo contiene códigos válidos, debe **eliminar del segundo vector los códigos encontrados en el primer vector**.

Ejemplo de programa de prueba:

```
int
main(void) {
    int lista1[] = {1, 3, 5, 7, 100, -1};
    int lista2[] = {3, 4, 5, 6, 7, -1};
    actualizarTareas(lista1, lista2);
    assert(lista1[0]==1 && lista1[1]==100 && lista1[2]==-1);
    assert(lista2[0]==4 && lista2[1]==6 && lista2[2]==-1);
    actualizarTareas(lista1, lista2);
    assert(lista1[0]==1 && lista1[1]==100 && lista1[2]==-1);
    assert(lista2[0]==4 && lista2[1]==6 && lista2[2]==-1);

    int lista3[] = {-1};
    actualizarTareas(lista1, lista3);
    assert(lista1[0]==1 && lista1[1]==100 && lista1[2]==-1);
    assert(lista3[0]==-1);

    int lista4[] = {4, 6, -1};
    actualizarTareas(lista2, lista4);
    assert(lista2[0]==-1);
    assert(lista4[0]==-1);

    puts("OK!");
    return 0;
}
```

Ejercicio 3 (3,5 puntos)

Implementar la función `int verificaMat` que recibe los siguientes parámetros:

- **`dim`**: `unsigned int` que debería ser mayor a cero
- **`mat`**: matriz de tipo `int`, de **`dim`** filas por **`dim`** columnas

La función debe **retornar 1 si en la misma están todos los enteros entre 1 y `dim` x `dim` inclusive, sin repetir ningún número**, y 0 en el caso contrario.

Por ejemplo si `dim` es 1, su único elemento debe ser 1, si `dim` es 2 tiene que tener los números 1, 2, 3 y 4 (en cualquier orden).