UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

Programación Orientada a Objetos 1

Práctica Calificada 2
Pregrado
2021-I
Profesor: Rubén Rivas Medina
Lab. 101

Indicaciones específicas:

- Esta evaluación contiene 9 páginas (incluyendo esta página) con 3 preguntas. El total de puntos son 20.
- El tiempo límite para la evaluación es 100 minutos.
- Cada pregunta deberá ser respondida en un solo archivo con el número de la pregunta. Por ejemplo:
  1. `p1.cpp y p1.h`
  2. `p2.cpp y p2.h`
  3. `p3.cpp y p3.h`
- Deberás subir estos archivos directamente a www.gradescope.com, o se puede crear un .zip que contenga todos ellos y subirlo.

Competencias y criterios de desempeño:

- Para los alumnos de la carrera de Ciencia de la Computación

  Aplica conocimientos de computación apropiados para la solución de problemas definidos y sus requerimientos en la disciplina del programa.(nivel 2)

  Diseña, implementa y evalúa soluciones a problemas complejos de computación.(nivel 2)

  Crea, selecciona, adapta y aplica técnicas, recursos y herramientas modernas para la práctica de la computación y comprende sus limitaciones.(nivel 2)

- Para los alumnos de las carreras de Ingeniería

  Capacidad para aplicar conocimientos de matemática.(nivel 2)

  Capacidad para diseñar un sistema, un componente o un proceso para satisfacer las necesidades deseadas dentro de restricciones realistas(nivel 2)

Calificación:

Tabla de puntos (sólo para uso del professor)

Question	Points	Score
1	7	
2	7	
3	6	
Total:	20	

1. (7 points) Escribir y diseñar una función (*sumar_numeros*) que retorne un vector de enteros y que permita leer un arreglo dinámico de enteros (n).

```
vector<int> sumar_numeros(int* arreglo,int n);
```

La función deberá retornar un vector que contenga solo 3 valores, el primero la suma de pares, la suma de impares y la suma de primos. Algunos ejemplos:

Ejemplo #1
Input Format

```
10
1 2 10 7 6 5 11 8 4 14
```

Output Format

```
44 24 25
```

Ejemplo #2
Input Format

```
6
1 2 10 7 6 5
```

Output Format

```
18 13 14
```

Ejemplo #3
Input Format

```
12
1 2 10 7 6 5 11 31 27 2 1 9
```

Output Format

```
20 92 58
```

La rúbrica para esta pregunta es:

Criterio	**Excelente**	**Adecuado**	**Mínimo**	**Insuficiente**
Algoritmo y codificación **(4 pts)**	Elabora un algoritmo preciso, definido y finito que da solución exacta a lo que el enunciado requiere. Utiliza arrays dinámicos y hace un buen uso de memoria: dimensiona, usa y libera memoria de manera adecuada al codificar el algoritmo y lo hace con el 100% de precisión. **(4pts)**	Elabora un algoritmo preciso, definido y finito que da solución al menos al 80 % de lo que el enunciado requiere. Utiliza arrays dinámicos y hace un buen uso de memoria: dimensiona, usa y libera memoria de manera adecuada al codificar el algoritmo y lo hace con al menos el 80% de precisión. **(3pts)**	Elabora un algoritmo preciso, definido y finito que da solución al menos al 65 % de lo que el enunciado requiere. Utiliza arrays dinámicos y hace un buen uso de memoria: dimensiona, usa y libera memoria de manera adecuada al codificar el algoritmo y lo hace con al menos el 65% de precisión. **(2pts)**	Elabora un algoritmo preciso, definido y finito que hace menos del 65 % de lo que el enunciado requiere. Utiliza arrays dinámicos y hace un buen uso de memoria: dimensiona, usa y libera memoria de manera adecuada al codificar el algoritmo y lo hace con menos el 65% de precisión. **(0 pts)**
Sintaxis y legibilidad **(1 pt)**	El algoritmo es correcto, y es codificado sin errores de sintaxis. El nombre de las variables y funciones son descriptivas. **(1pts)**	El algoritmo es correcto, y es codificado con algunos errores de sintaxis, pero que no afectan el resultado de manera significativa. El nombre de las variables y funciones son descriptivas. **(0.75pts)**	El algoritmo es correcto, y es codificado con algunos errores de sintaxis, que afectan el resultado de manera mínima, o el nombre de las variables y funciones no son descriptivas. **(0.5pts)**	El algoritmo es incorrecto o es codificado con errores de sintaxis, que afectan el resultado de manera significativa. El nombre de las variables y funciones no son descriptivas. **(0pts)**
Optimización de código **(2 pt)**	El código es óptimo y eficiente **(2pts)**	El código es óptimo en al menos el 80% **(1.5pts)**	El código es óptimo en al menos 65% **(1pts)**	El código es redundante y/o no es óptimo**(0pts)**

2. (7 points) Escribir y diseñar la función (*esta_rodeado*) que reciba una matriz cuadrada de lado (n) la cual almacenará los valores enteros 0s y 1s, en donde los 1s representarán los bordes y la función deberá detectar si los 0s no tocan ($true$) o tocan ($false$) los bordes.

```
bool esta_rodeado(int** matriz, int n);
```

Algunos ejemplos:

Ejemplo #1
Input Format

```
10
1 1 1 1 1 1 1 1 1 1
1 1 1 0 0 0 1 1 1 1
1 1 1 0 0 0 1 1 1 1
1 1 1 0 0 0 0 1 1 1
1 1 1 0 0 0 0 0 1 1
1 1 0 0 0 0 0 1 1 1
1 1 0 0 0 0 0 0 0 1
1 1 0 1 0 0 0 0 1 1
1 1 0 1 0 0 0 0 0 1
1 1 1 1 1 1 1 1 1 1
```

Output Format

```
true
```

Ejemplo #2
Input Format

```
4
1 1 0 1
1 0 0 1
1 0 1 1
1 1 1 1
```

Output Format

```
false
```

Ejemplo #3
Input Format

```
12
1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1
```

```
1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 0 0 0 1 1
1 1 1 1 1 1 1 0 0 0 0 1
1 1 1 1 1 1 1 0 1 0 0 1
1 1 1 1 1 1 1 1 1 1 0 0
```

Output Format

```
false
```

La rúbrica para esta pregunta es:

Criterio	**Excelente**	**Adecuado**	**Mínimo**	**Insuficiente**
Algoritmo y codificación **(4 pts)**	Elabora un algoritmo preciso, definido y finito que da solución exacta a lo que el enunciado requiere. Utiliza matrices dinámicas y hace un buen uso de memoria: dimensiona, usa y libera memoria de manera adecuada al codificar el algoritmo y lo hace con el 100% de precisión. **(4pts)**	Elabora un algoritmo preciso, definido y finito que da solución al menos al 80 % de lo que el enunciado requiere. Utiliza matrices dinámicas y hace un buen uso de memoria: dimensiona, usa y libera memoria de manera adecuada al codificar el algoritmo y lo hace con al menos el 80% de precisión. **(3pts)**	Elabora un algoritmo preciso, definido y finito que da solución al menos al 65 % de lo que el enunciado requiere. Utiliza matrices dinámicas y hace un buen uso de memoria: dimensiona, usa y libera memoria de manera adecuada al codificar el algoritmo y lo hace con al menos el 65% de precisión. **(2pts)**	Elabora un algoritmo preciso, definido y finito que hace menos del 65 % de lo que el enunciado requiere. Utiliza matrices dinámicas y hace un buen uso de memoria: dimensiona, usa y libera memoria de manera adecuada al codificar el algoritmo y lo hace con menos el 65% de precisión. **(0 pts)**
Sintaxis y legibilidad **(1 pt)**	El algoritmo es correcto, y es codificado sin errores de sintaxis. El nombre de las variables y funciones son descriptivas. **(1pts)**	El algoritmo es correcto, y es codificado con algunos errores de sintaxis, pero que no afectan el resultado de manera significativa. El nombre de las variables y funciones son descriptivas. **(0.75pts)**	El algoritmo es correcto, y es codificado con algunos errores de sintaxis, que afectan el resultado de manera mínima, o el nombre de las variables y funciones no son descriptivas. **(0.5pts)**	El algoritmo es incorrecto o es codificado con errores de sintaxis, que afectan el resultado de manera significativa. El nombre de las variables y funciones no son descriptivas. **(0pts)**
Optimización de código **(2 pt)**	El código es óptimo y eficiente **(2pts)**	El código es óptimo en al menos el 80% **(1.5pts)**	El código es óptimo en al menos 65% **(1pts)**	El código es redundante y/o no es óptimo**(0pts)**

3. (6 points) Escribir la función (*obtener_repetidos*) que tenga como parámetro un vector de numeros enteros cuyo rango es desde 1 hasta 60 y un parámetro (*repeat*) un número entero que cuente las repeticiones la función debe generar un vector ordenado con todos aquellos números que se repitan las veces que se solicitan.

```
vector<int> obtener_repetidos(vector<int>& vec, int repeat);
```

Algunos ejemplos: **Ejemplo #1**
Input Format

```
9
1 2 5 7 8 5 4 8 8
2
```

Output Format

```
5
```

Ejemplo #2
Input Format

```
20
1 3 6 5 4 12 31 20 11 2 6 3 4 5 11 59 21 4 50 11
3
```

Output Format

```
4 11
```

Ejemplo #3
Input Format

```
16
1 2 10 7 6 5 11 15 10 3 1 9 13 20 14 4
1
```

Output Format

```
2 3 4 5 6 7 9 11 13 14 15 20
```

La rúbrica para esta pregunta es:

Criterio	**Excelente**	**Adecuado**	**Mínimo**	**Insuficiente**
Algoritmo y codificación **(3 pts)**	Elabora un algoritmo preciso, definido y finito que da solución exacta a lo que el enunciado requiere. **Utiliza vectores y sus métodos** al codificar el algoritmo y lo hace con el 100% de precisión. **(4pts)**	Elabora un algoritmo preciso, definido y finito que da solución al menos al 80 % de lo que el enunciado requiere. **Utiliza vectores y sus métodos** al codificar el algoritmo y lo hace con al menos el 80% de precisión. **(2pts)**	Elabora un algoritmo preciso, definido y finito que da solución al menos al 65 % de lo que el enunciado requiere. **Utiliza vectores** al codificar el algoritmo y lo hace con al menos el 65% de precisión. **(1pts)**	Elabora un algoritmo preciso, definido y finito que hace menos del 65 % de lo que el enunciado requiere. **Utiliza vectores y sus métodos** al codificar el algoritmo y lo hace con menos el 65% de precisión. **(0 pts)**
Sintaxis y legibilidad **(1 pt)**	El algoritmo es correcto, y es codificado sin errores de sintaxis. El nombre de las variables y funciones son descriptivas. **(1pts)**	El algoritmo es correcto, y es codificado con algunos errores de sintaxis, pero que no afectan el resultado de manera significativa. El nombre de las variables y funciones son descriptivas. **(0.75pts)**	El algoritmo es correcto, y es codificado con algunos errores de sintaxis, que afectan el resultado de manera mínima, o el nombre de las variables y funciones no son descriptivas. **(0.5pts)**	El algoritmo es incorrecto o es codificado con errores de sintaxis, que afectan el resultado de manera significativa. El nombre de las variables y funciones no son descriptivas. **(0pts)**
Optimización de código **(2 pt)**	El código es óptimo y eficiente **(2pts)**	El código es óptimo en al menos el 80% **(1.5pts)**	El código es óptimo en al menos 65% **(1pts)**	El código es redundante y/o no es óptimo**(0pts)**