UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

Programación Orientada a Objetos 1

Práctica Calificada 2
Pregrado
2021-I
Profesor: Rubén Rivas Medina
Lab. 106

Indicaciones específicas:

- Esta evaluación contiene 8 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 (*obtener_indices*) que retorne un vector de enteros y que permita leer un arreglo dinámico con valores enteros de tamaño (n) y un carácter (*opcion*) que defina las siguientes opciones:

   - p par
   - i impar
   - r primo

```
vector<int> obtener_indices(int* arreglo,int n,char opcion);
```

La función deberá retornar un vector que contenga con los subíndices de los valores de acuerdo al parametro opción, por ejemplo: si se elige la opción **p** debera generar un vector con todos los subíndices de los valores que tengan valor par. si se elige la opción **i** debera generar un vector con todos los subíndices de los valores que tengan valor impar y si se elige la opción **r** debera generar un vector con todos los subíndices de los valores que tengan valor primos. Algunos ejemplos:

Ejemplo #1
Input Format

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

Output Format

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

Ejemplo #2
Input Format

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

Output Format

```
0 3 5
```

Ejemplo #3
Input Format

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

Output Format

```
1 3 5 6 7 9
```

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 una función (*generar_matriz_organizada*) que retorne una matriz cuadrada de enteros de lado (n) y que permita leer un arreglo dinámico con valores enteros de tamaño ($n \cdot n$).

```
int** generar_matriz_organizada(int* arreglo, int n);
```

La función debe tomar cada valor del arreglo y calcular su posición en la matriz usando las formulas: ($i = valor/n$) y ($j = valor\%n$) y ubicar el valor en la posición correspondiente.

La matriz inicialmente debe tener sus valores en CERO.

Si la posición estuviera fuera del rango de posiciones válidas de la matriz entonces el valor se descarta.

Si anteriormente otro valor ocupará esa posición el valor deberá ser sobreescrito.

Algunos ejemplos:

Ejemplo #1
Input Format

```
3
1 2 5 7 6 5 4 8
```

Output Format

```
0 1 2
0 4 5
6 7 8
```

Ejemplo #2
Input Format

```
2
1 2 6 5
```

Output Format

```
0 1
2 0
```

Ejemplo #3
Input Format

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

Output Format

```
0  1  2  3
4  5  6  7
0  9 10 11
0 13 14 15
```

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 un programa que lea (n) valores enteros y que los almacene en un vector, modificar el vector para que solo guarde en forma ordenada los valores sin repetirlos.

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

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

   Output Format

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

   Ejemplo #2
   Input Format

```
20
1 2 6 5 4 12 31 20 11 2 6 3 4 5 11 59 21 22 50 11
```

   Output Format

```
1 2 3 4 5 6 11 12 20 21 22 31 50 59
```

   Ejemplo #3
   Input Format

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

   Output Format

```
1 2 3 4 5 6 7 9 10 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)**