UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

Programación Orientada a Objetos 1

Práctica Calificada 1
Pregrado
2020-II
Profesora: María Hilda Bermejo
Lab. 1.10

Indicaciones específicas:

- Esta evaluación contiene 12 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 y tu código de estudiante. Por ejemplo:
  1. `p1_2020010202.cpp`
  2. `p2_2020010202.cpp`
  3. `p3_2020010202.cpp`
- Estos archivos deben estar en una carpeta llamada PC1. Una vez que termines de responder a las preguntas. Comprime la carpeta PC1, lo cual generará el archivo PC1.Zip
- Deberás subir el archivo **PC1.zip** directamente a www.gradescope.com, uno en cada ejercicio.

Competencias:

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

  Aplicar conocimientos de computación y de matemáticas apropiadas para la disciplina. (Evaluar)

  Analizar problemas e identificar y definir los requerimientos computacionales apropiados para su solución.(Usar)

  Utilizar técnicas y herramientas actuales necesarias para la práctica de la computación. (Usar)

- Para los alumnos de las carreras de Ingeniería

  Capacidad de aplicar conocimientos de matemáticas (nivel 3)

  Capacidad de aplicar conocimientos de ingeniería(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	6	
2	7	
3	7	
Total:	20	

1. (6 points) **Capicúas**
   Desarrolle un programa que permita imprimir todos los números que sean capicuas, desde el 11 hasta un número que se ingrese como límite.

   - El programa debe verificar que el número que se ingrese como límite sea mayor o igual a 11. Si el usuario ingresa un número que no es mayor o igual a 11, debe volver a pedir el dato.
   - Note que el límite puede ser un número de varios dígitos.
   - Un número es capicúa, si se lee el mismo número cuando se lee de izquierda a derecha y cuando se lee de derecha a izquierda. Por ejemplo: son números capicúas el 77677, 838, 9889

   Algunos ejemplos de diálogo de este programa serían:

Listing 1: Ejemplo 1

```
Numero > 10  : 8
Numero > 10  : -3
Numero > 10  : 10
Numero > 10  : 99

11
22
33
44
55
66
77
88
99
```

Listing 2: Ejemplo 1

```
Numero > 10  : 879

11
22
33
44
55
66
77
88
99
101
111
121
```

```
131
141
151
161
171
181
191
202
212
222
232
242
252
262
272
282
292
303
313
323
333
343
353
363
373
383
393
404
414
424
434
444
454
464
474
484
494
505
515
525
535
545
555
565
575
```

```
585
595
606
616
626
636
646
656
666
676
686
696
707
717
727
737
747
757
767
777
787
797
808
818
828
838
848
858
868
878
```

Listing 3: Ejemplo 1

```
Numero > 10  : 11

11
```

Listing 4: Ejemplo 1

```
Numero > 10  : 3000

11
22
33
44
55
```

```
66
77
88
99
101
111
121
131
141
151
161
171
181
191
202
212
222
232
242
252
262
272
282
292
303
313
323
333
343
353
363
373
383
393
404
414
424
434
444
454
464
474
484
494
505
```

```
515
525
535
545
555
565
575
585
595
606
616
626
636
646
656
666
676
686
696
707
717
727
737
747
757
767
777
787
797
808
818
828
838
848
858
868
878
888
898
909
919
929
939
949
959
```

```
969
979
989
999
1001
1111
1221
1331
1441
1551
1661
1771
1881
1991
2002
2112
2222
2332
2442
2552
2662
2772
2882
2992
```

La rúbrica para esta pregunta es:

Criterio	Logrado	Parcialmente Logrado	No Logrado
Algoritmo y Código	El algoritmo y código es preciso y finito y hace exactamente lo que el enunciado requiere. (`3pts`)	Es preciso, finito y hace la mitad o más de lo que el enunciado requiere. (`2pts`)	Hace menos de la mitad de lo que el enunciado requiere (`0pts`).
Sintaxis	No existen errores sintácticos o de compilación (`2pts`)	Existen algunos errores sintácticos o de compilación. (`1pts`).	El código no compila (`0pts`).
Optimizacion	El código es óptimo y eficiente (`1pts`)	El codigo es optimizable en algunas partes (`0.5pts`).	El codigo es redundante y/o no es optimo (`0pts`).

2. (7 points) **Temperaturas**

Una estación metereológica de Senamhi, desea contar con un programa que permita registrar las temperaturas durante cierta cantidad de días, para luego tener un reporte de la cantidad de días: frios, templados y calurosos que hubo en ese periodo de tiempo. Asimismo desea saber cuál fue la temperatura promedio y la temperatura máxima.

- El programa tiene como dato la cantidad de días
- El rango de temperaturas para poder clasificar se muestra en la tabla.

Tipo de Día	Rango de temperaturas
Frio	hasta 15 grados
Templado	de mas de 15 grados hasta 21 grados
Caluroso	mas de 21 grados

Algunos ejemplos de diálogo de este programa serían:

Listing 5: Ejemplo 1

```
Numero de dias : 5
Temperatura del dia 1 : 23
Temperatura del dia 2 : 12
Temperatura del dia 3 : 18
Temperatura del dia 4 : 17
Temperatura del dia 5 : 20
Reporte
Temperatura Promedio : 18
Temperatura Maxima   : 23
Dias Frios     [temp menor a 16 grados ] : 1
Dias Templados [temp >15 - 21 grados]    : 3
Dias Calurosos [temp mayor a 21]         : 1
```

Algunos ejemplos de diálogo de este programa serían:

Listing 6: Ejemplo 1

```
Numero de dias : 10
Temperatura del dia 1 : 12
Temperatura del dia 2 : 21
Temperatura del dia 3 : 24
Temperatura del dia 4 : 19
Temperatura del dia 5 : 9
Temperatura del dia 6 : 22
Temperatura del dia 7 : 19
Temperatura del dia 8 : 16
Temperatura del dia 9 : 14
Temperatura del dia 10 : 15
Reporte
```

```
Temperatura Promedio : 17.1
Temperatura Maxima   : 24
Dias Frios     [temp menor a 16 grados ] : 4
Dias Templados [temp >15 - 21 grados]    : 4
Dias Calurosos [temp mayor a 21]         : 2
```

La rúbrica para esta pregunta es:

Criterio	Logrado	Parcialmente Logrado	No Logrado
Algoritmo y Código	El algoritmo y código es preciso y finito y hace exactamente lo que el enunciado requiere. (`4pts`)	Es preciso, finito y hace la mitad o más de lo que el enunciado requiere. (`2pts`)	Hace menos de la mitad de lo que el enunciado requiere (`0pts`).
Sintaxis	No existen errores sintácticos o de compilación (`2pts`)	Existen algunos errores sintácticos o de compilación. (`1pts`).	El código no compila (`0pts`).
Optimizacion	El código es óptimo y eficiente (`1pts`)	El codigo es optimizable en algunas partes (`0.5pts`).	El codigo es redundante y/o no es optimo (`0pts`).

3. (7 points) **Números primos**
Desarrolle un programa que permita leer un número entero mayor a 10 y el programa imprima el siguiente número primo y el número primo inmediato anterior.

   - El programa debe verificar el ingreso del número, si este no es mayor a 10, volverá a solicitar el número.
   - Recuerde que un número es primo si solo es divisible entre 1 y en mismo número.

A continuación se le presenta el código correspondiente a la función main(), y se pide que Ud, escriba el programa completo, implementando la función **misPrimos**. Que está enviado en sus dos últimos parámetros, **las direcciones de memoria** de las variables: **anteriorPrimo** y **sigPrimo**, en donde la función dejará los datos correspondientes.
Adicionalmente a esta función, Ud. puede diseñar e implementar las funciones que considere necesarias.

Listing 7: Ejemplo 1

```
#include <iostream>
using namespace std;


int main()
{
  long long int num=0;
  long long int anteriorPrimo=0, sigPrimo=0;
  do {
    cout << "Numero␣[mayor␣a␣10␣]␣:␣";
    cin>>num;
  }while(num<=10);

  misPrimos(num, &anteriorPrimo, &sigPrimo);
  cout<<"El␣siguiente␣numero␣primo␣es:␣" << sigPrimo <<"\n";
  cout<<"El␣anterior␣numero␣primo␣es:"<<anteriorPrimo<<"\n";
  return 0;
}
```

Algunos ejemplos de diálogo de este programa serían:

Listing 8: Ejemplo 1

```
Numero [mayor a 10 ] : 13
El siguiente numero primo es : 17
El anterior numero primo es  : 11
```

Listing 9: Ejemplo 1

```
Numero [mayor a 10 ] : 113
```

```
El siguiente numero primo es : 127
El anterior numero primo es  : 109
```

Listing 10: Ejemplo 1

```
Numero [mayor a 10 ] : 3413
El siguiente numero primo es : 3433
El anterior numero primo es  : 3407
```

La rúbrica para esta pregunta es:

Criterio	Logrado	Parcialmente Logrado	No Logrado
Código	Ha implementado funciones y punteros en forma correcta y lógica (`4pts`)	Existen algunos errores menores en la implementación (`2pts`)	El diseño y la implementacion del código no son correctos (`0pts`).
Sintaxis	No existen errores sintácticos o de compilación (`2pts`)	Existen algunos errores sintácticos o de compilación. (`1pts`).	El código no compila (`0pts`).
Optimizacion	El código es óptimo y eficiente (`1pts`)	El código es optimizable en algunas partes (`0.5pts`).	El código es redundante y/o no es óptimo (`0pts`).