Modelos de Ejercicios y Exámenes de Estructura de Datos I

Ing. Mario López Winnipeg

Creado por:

Univ. Willy Vargas Méndez

Facultad de Ingeniería en Ciencias de la Computación y Telecomunicaciones

Universidad Autónoma Gabriel René Moreno

Para cualquier consulta o reclamo escribir al correo: wamdev01@gmail.com

Los siguientes problemas y ejercicios son recopilación de tareas y exámenes la materia estructura de datos I impartida por el Ing. Mario López Winnipeg

**Tema Clase Memoria**

1. Ejecute las siguientes líneas de código y muestre como queda la memoria simulada.  
     
   Ejemplo Si mario milton lopez winnipeg con registro 200912421 tendría que elaborar la tarea estas serían las líneas de código.

d1 = ObjMem.New\_espacio("mario,milton,lopez,winnipeg");    
d2 = ObjMem.New\_espacio("mario,milton,lopez,winnipeg");  
ObjMem.poner\_dato(d1,"->winnipeg",912421);  
ObjMem.mostrar();

RECORDARLES QUE EL METODO MOSTRAR DE LA CLASE CMEMORIA MUESTRA EN MODO CONSOLA TODO EL CONTENIDO DEL VECTOR Y LO QUE EXISTE EN EL ATRIBUTO LIBRE.

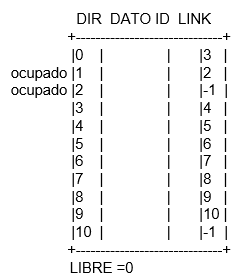
1. Ejecute las siguientes líneas de código y muestre como queda la memoria simulada.  
     
   Ejemplo Si mario milton lopez winnipeg con registro 200912421 tendría que elaborar la tarea estas serían las líneas de código.

d1 = ObjMem.New\_espacio("mario,milton,lopez,winnipeg");    
d2 = ObjMem.New\_espacio("mario,milton,lopez,winnipeg");  
ObjMem.poner\_dato(d1,"->winnipeg",912421);  
ObjMem.delete\_espacio(d1);  
d1 = ObjMem.New\_espacio("mario,milton,lopez,winnipeg");  
ObjMem.poner\_dato(d1,"->winnipeg",

ObjMem.obtenerdato(d1,"->winnipeg")+1);  
ObjMem.mostrar();

RECORDARLES QUE EL METODO MOSTRAR DE LA CLASE CMEMORIA MUESTRA EN MODO CONSOLA TODO EL CONTENIDO DEL VECTOR Y LO QUE EXISTE EN EL ATRIBUTO LIBRE.

1. Modificar el método **mostrar** perteneciente a la **clase Memoria** para que muestre todos sus datos de la siguiente forma.

****

**Tema TDA Lista**

1. Ejecute las siguientes líneas de código y muestre como queda la memoria simulada.

A = M.New\_espacio(“uno”)

B = M.New\_espacio(“dos”)

C = M.new\_espacio(“tres”)

D = M.new\_espacio(“cuatro”)

M.delete\_espacio(B)

M.delete\_espacio(D)

E = M.new\_espacio(“mario,milton,lopez,winnipeg”)//Aquí su nombre

L.inserta(L.fin(),777)

M.delete\_espacio(A)

L.inserta(L.primero(),666 )

M.mostrar()

**Tomar en cuenta que la instancia del TDA Lista “L” usa una Memoria Simulada para guardar sus datos.**

1. Implemente el código del **método inserta** perteneciente a la **clase lista** implementada con punteros cuyos nodos tendrán la siguiente **estructura (ele, sig)** y la clase lista tendrá como únicos atributos lo siguiente **puntero\_a\_lista** de tipo puntero a nodo y **nro\_de\_elementos** de tipo numérico.
2. Modifique el código del **método suprime** de la **clase lista** implementada con punteros considerando los siguientes cambios el NODO tiene la siguiente **estructura (Datillo, próximo)** y los atributos de la clase son **PunteroD y Cantidad**, donde **punteroD** es el puntero a Nodo y **Cantidad** indica la cantidad de elementos en la lista.

**Tema TDA Conjunto**

1. Implementar el procedimiento denominado:

**COMPLEMENTO\_DE\_INTERSECCION**(**E** A, B CONJUNTO, **ES** C CONJUNTO)

Considerando que A y B son parámetros de entrada por valor y C es parámetro por referencia.

Traducido a C++

**void** complemento\_de\_interseccion(**Conjunto** a,**Conjunto** b,

**Conjunto** &c);

**A** = {22, 44, 100, 66, 500}

**B** = {44, 33, 77, 22}

**A ∩ B** = {22, 44}

**C** = **Complemento**(A ∩ B) = {100, 66, 500, 33, 77}

La clase Conjunto esta implementada con punteros de la siguiente manera:

**class** Conjunto{

**private:**

Nodo\* ptrconj;

**int** cant;

**public:**

Conjunto();

**boolean** vacio();

**int** cardinal();

**int** ordinal(**int** e);

**void** inserta(**int** e);

**void** suprime(**int** e);

**boolean** pertenece(**int** e);

};

1. Implementar el método denominado **ordinal** perteneciente a la **clase Conjunto**.

La clase Conjunto esta implementada con una Lista de la siguiente manera:

**class** Conjunto{

**private:**

Lista elem;

**public:**

Conjunto();

**boolean** vacio();

**int** cardinal();

**int** ordinal(**int** e);

**void** inserta(**int** e);

**void** suprime(**int** e);

**boolean** pertenece(**int** e);

**int** muestrea();

};

La clase Lista esta implementada con simulación de memoria de la siguiente manera:

**typedef int** direccion;

**class** Lista{

**private:**

direccion ptrelementos;

**int** Longitud;

**public:**

Lista();

direccion fin();

direccion primero();

direccion siguiente(direccion dir);

direccion anterior(direccion dir);

**boolean** vacia();

**int** recupera(direccion dir);

**int** longitud();

**void** inserta(direccion dir, **int** e);

**void** suprime(direccion dir);

**void** modifica(direccion dir, **int** e);

};

**Tema TDA Polinomio**

1. Implemente el método denominado **derivada** perteneciente a la **clase Polinomio**, la cual esta implementada de la siguiente forma:

**class** Polinomio {

**private:**

Lista Poli;

**public:**

Polinomio();

**boolean** esCero();

**void** poner\_termino(**int** coef, **int** exp);

**int** coeficiente(**int** exp);

**int** exponente(**int** nro\_termino);

**int** grado();

**void** suma(Polinomio p1, Polinomio &p2);

**void** resta(Polinomio p1, Polinomio &p2);

**int** numero\_terminos();

**void** derivada();

};

**Tomar en cuenta que la lista de la clase polinomio guarda sus datos de la siguiente forma:**

Polinomio =

Lista = {7, 9, 1, 5, 3, 4}

Lista = {exp, coef, exp, coef, exp, coef}

1. Implemente el método denominado **poner\_termino** perteneciente a la **clase polinomio** la cual esta implementada de la siguiente forma:

**struct** Nodo {

**int** ex;

**int** co;

Nodo\* prox;

};

**class** Polinomio {

**private:**

Nodo\* ptr;

**int** nt;

**public:**

Polinomio();

**boolean** esCero();

**void** poner\_termino(**int** coef, **int** exp);

**int** coeficiente(**int** exp);

**int** exponente(**int** nro\_termino);

**int** grado();

**void** suma(Polinomio p1, Polinomio &p2);

**void** resta(Polinomio p1, Polinomio &p2);

**int** numero\_terminos();

**void** derivada();

};

1. Implemente el procedimiento denominado:

**Area**(FX: Polinomio; A, B: Real)

Traducido a c++

**float** area(Polinomio fx, **float** a**, float** b);

**NO PUEDE MODIFICAR NADA DE LAS CLASE POLINOMIO LA CUAL PUDE USAR LOS SIGUIENTES METODOS:**

Polinomio();

**boolean** esCero();

**void** poner\_termino(**int** coef, **int** exp);

**int** coeficiente(**int** exp);

**int** exponente(**int** nro\_termino);

**int** grado();

**void** suma(Polinomio p1, Polinomio &p2);

**void** resta(Polinomio p1, Polinomio &p2);

**int** numero\_terminos();

**void** derivada();

1. Implemente el procedimiento denominada:

**void** punto\_minimo(Polinomio &fx, **float** a, **float** b);

dicho procedimiento deberá mostrar por pantalla por lo menos una

coordenada de los **puntos mínimos** de una **función fx**, en el intervalo **a** y **b cerrado**.

**Tema TDA Matriz Dispersa**

1. Implemente el **método poner** perteneciente a la **clase MatrizDispersa** que esta **implementada con punteros**, considerando que la clase tiene los siguientes atributos:

**private:**

**int** nt;

NodoF\* ptrFil;

**int** rep, nf, nc;

**struct** NodoF {

**int** fil;

NodoF\* sigF;

NodoF\* antF;

NodoC\* ptrCol;

};

**struct** NodoC {

**int** col;

**int** dato;

NodoC\* sigCol;

};

1. Implemente el **método** definir\_valor\_repetido(**int** valor) perteneciente a la **clase MatrizDispersa** que esta implementada con punteros que tiene los siguientes atributos:

**private:**

**int** nt;

Nodo\* ptrMat;

**int** rep, dimf, dimc;

**struct** Nodo {

**int** fi, co, da;

Nodo\* next;

};

**Tema TDA Pila**

1. Implemente los **métodos poner y sacar** pertenecientes a la **clase pila** que tiene como único atributo **L de tipo lista** donde siempre el último elemento ingresado esta al final, considere que no existe un método en la clase lista que inserte un elemento después del último.
2. Implementar los **métodos meter y sacar** pertenecientes a la **clase pila**, considere que el usuario que utilice las pilas manipulara valores enteros, sin embargo, en su implementación la misma tiene las siguientes condiciones:

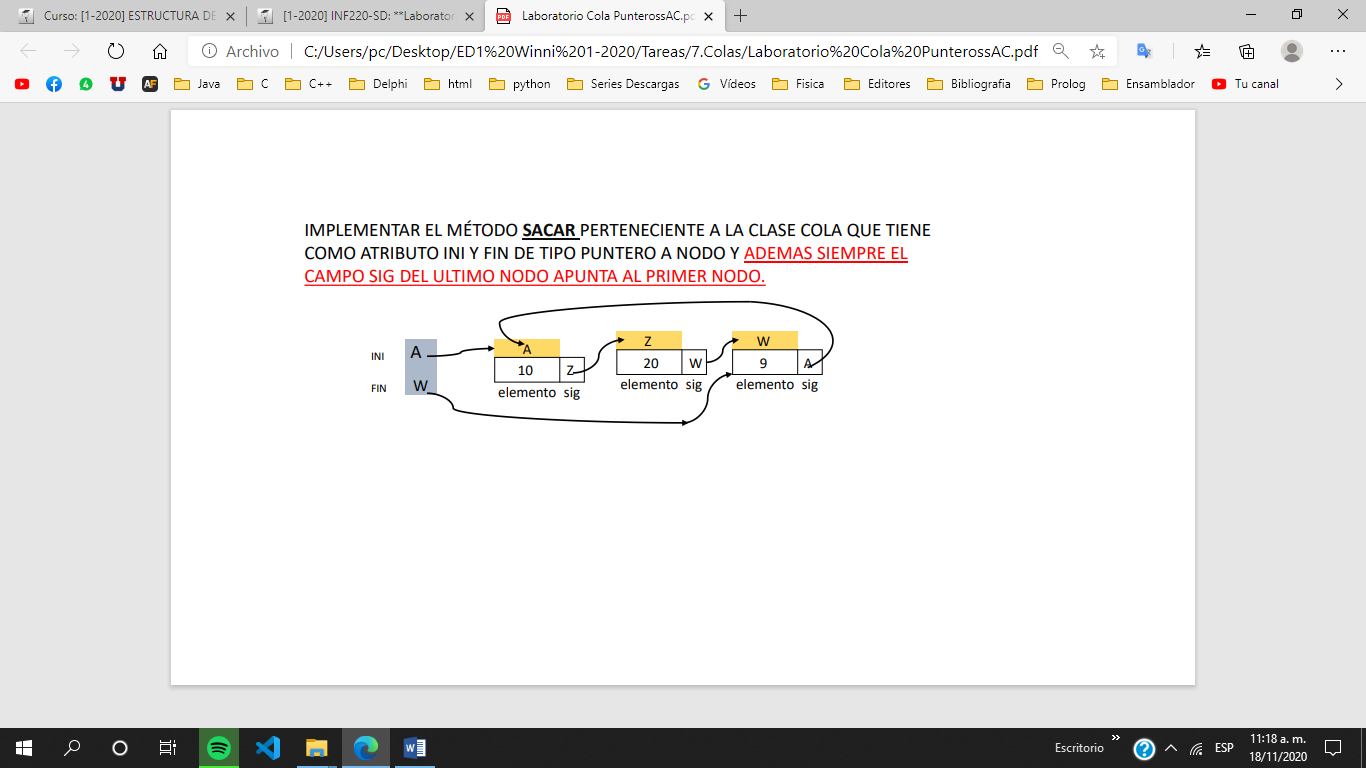
* Tiene un solo atributo K de tipo Cadena
* Tiene una capacidad máxima para almacenar 10 elementos

**Tema TDA Cola**

1. Implementar el **método sacar** perteneciente a la **clase cola** que tiene

como atributo **ini** y **fin** de tipo puntero a nodo y además siempre el

campo **sig** del ultimo nodo apunta al primer nodo.



1. Implemente el código del **método** **poner** y **poner\_frente** perteneciente a la clase **Dicola** que tiene como único atributo **K** de tipo Lista, la lista esta implementada con punteros.

**Modelos de Examen**



**1er Parcial**

1. Dado es siguiente estado de memoria a su derecha, dados los objetos **L de tipo Lista**, **C de tipo Conjunto** y **P de tipo Polinomio** indique en qué estado quedará la memoria después de ejecutarse las siguientes líneas:

delete\_espacio(1)

L.inserta(L.primero(), X)

L.inserta(L.primero(), Y)

delete\_espacio(2)

C.inserta(Z)

P.poner\_termino(2, 5)

delete\_espacio(7)

L.inserta(L.fin(), W)

**Todos los objetos están usando la misma memoria simulada**

**Considere los valores de X, Y, Z y W según su número de registro:**

Ejm:

**Reg :** 324067098

X = 24

Y = 06

Z = 70

W = 98

1. Implemente el **método poner\_termino** perteneciente a la **clase Polinomio** considerando la clase esta implementada de la siguiente forma:

**class** Polinomio {

**private:**

Lista K;

**public:**

Polinomio();

**boolean** esCero();

**void** poner\_termino(**int** coef, **int** exp);

**int** coeficiente(**int** exp);

**int** exponente(**int** nro\_termino);

**int** grado();

**void** suma(Polinomio p1, Polinomio &p2);

**void** resta(Polinomio p1, Polinomio &p2);

**int** numero\_terminos();

**void** derivada();

};

**Los Datos se guardan de la siguiente forma en la lista**

Polinomio =

Lista(K) = {3, 9, 1, 5, 7, 4, 7}

Lista = {exp, coef, exp, coef, exp, coef, gradoMaximo}

**Nota: Pueden usar libremente los métodos de la clase Polinomio, no pueden implementar ninguno nuevo.**

1. Implemente el **método poner\_termino** perteneciente a la **clase Polinomio** considerando la clase esta implementada de la siguiente forma:

**class** Polinomio {

**private:**

Lista K;

**public:**

Polinomio();

**boolean** esCero();

**void** poner\_termino(**int** coef, **int** exp);

**int** coeficiente(**int** exp);

**int** exponente(**int** nro\_termino);

**int** grado();

**void** suma(Polinomio p1, Polinomio &p2);

**void** resta(Polinomio p1, Polinomio &p2);

**int** numero\_terminos();

**void** derivada();

};

**Los Datos se guardan de la siguiente forma en la lista**

Polinomio =

Lista(K) = {7, 9, 3, 5, 1, 4, 7}

Lista = {gradoMaximo, coef, exp, coef, exp, coef, exp}

**Nota: Pueden usar libremente los métodos de la clase Polinomio, no pueden implementar ninguno nuevo.**

1. Implementar el procedimiento denominado **intersección** que tiene 3 parámetros por valor y 1 por referencia, considerando q la **clase conjunto** tiene un solo atributo **K** de tipo **Lista** **implementada con punteros**.

**void** intersección(Conjunto A, Conjunto B, Conjunto C,

Conjunto &CR);

**Ejm:**

**A = {**3, 4, 5, 6**}**

**B = {**3, 4, 10, 20**}**

**C = {**3, 10, 11, 20, 4**}**

**CR = (A ∩ B ∩ C) = {**3, 4**}**

**NOTA: Todos los métodos de la clase Conjunto y la clase Lista ya están implementados y funcionan correctamente, NO PUEDE ADICIONAR UN NUEVO METODO A LA CLASE**

**METODOS CLASE LISTA:** [ListaMetodos](#ListaMetodos)

**METODOS CLASE CONJUNTO:** [ConjuntoMetodos](#ConjuntoMetodos)

1. Implementar el método denominado **definir\_valor\_repetido** perteneciente a la **clase Matriz Dispersa** que esta implementada con vectores bajo el formato coordenado sin compresión.

**2do Parcial**

1. Implemente los métodos **poner, poner\_frente** **y sacar** de la clase **Dicola** considerando que la clase **Dicola** tiene como único atributo **K** de tipo **Pila**, pila que está **implementada con vectores**.

**NOTA:** Puede usar los demás métodos de la clase Dicola y todos los métodos de la clase Pila.

**METODOS CLASE DICOLA:** [DicolaMetodos](#DicolaMetodos)

**METODOS CLASE PILA:** [PilaMetodos](#PilaMetodos)

1. Implemente los métodos **poner, poner\_frente** **y sacar** de la **clase** **Dicola** considerando que la clase **Dicola** tiene como único atributo **K** de tipo **Cola**, cola que está **implementada con vectores**.

**NOTA:** Puede usar los demás métodos de la clase Dicola y todos los métodos de la clase Cola.

**METODOS CLASE DICOLA:** [DicolaMetodos](#DicolaMetodos)

**METODOS CLASE COLA:** [ColaMetodos](#ColaMetodos)

1. Implemente los métodos **vacía y primero** de la **clase** **ColadePrioridad** considerando que las colas usadas en el **vector VC** están **implementada con punteros**.

**NOTA:** Puede usar los demás métodos de la clase ColadePrioridad y todos los métodos de la clase Cola.

**METODOS CLASE COLADEPRIORIDAD:** [ColadePrioridadadMetodos](#ColadePrioridadadMetodos)

**METODOS CLASE COLA:** [ColaMetodos](#ColaMetodos)

1. Implemente el método denominado **void** primero(**int** &valor);

de la **clase** **ColadePrioridad**, considerando que no puede usar recursividad y que las colas del **vector VC** están **implementadas con punteros**.

**NOTA:** Puede usar los demás métodos de la clase ColadePrioridad y todos los métodos de la clase Cola.

**METODOS CLASE COLADEPRIORIDAD:** [ColadePrioridadadMetodos](#ColadePrioridadadMetodos)

**METODOS CLASE COLA:** [ColaMetodos](#ColaMetodos)

1. Implemente el método **suprime** perteneciente a la **clase** **Lista** que esta implementada de la siguiente forma:

**struct** Nodo{

Nodo\* ant;

**int** elem;

Nodo\* sig;

};

**class** Lista{

**private:**

Nodo\* Ptr\_lista;

**int** Longitud;

**public:**

Lista();

Nodo\* fin();

Nodo\* primero();

Nodo\* siguiente(Nodo\* P);

Nodo\* anterior(Nodo\* P);

**boolean** vacia();

**int** recupera(Nodo\* P);

**int** longitud();

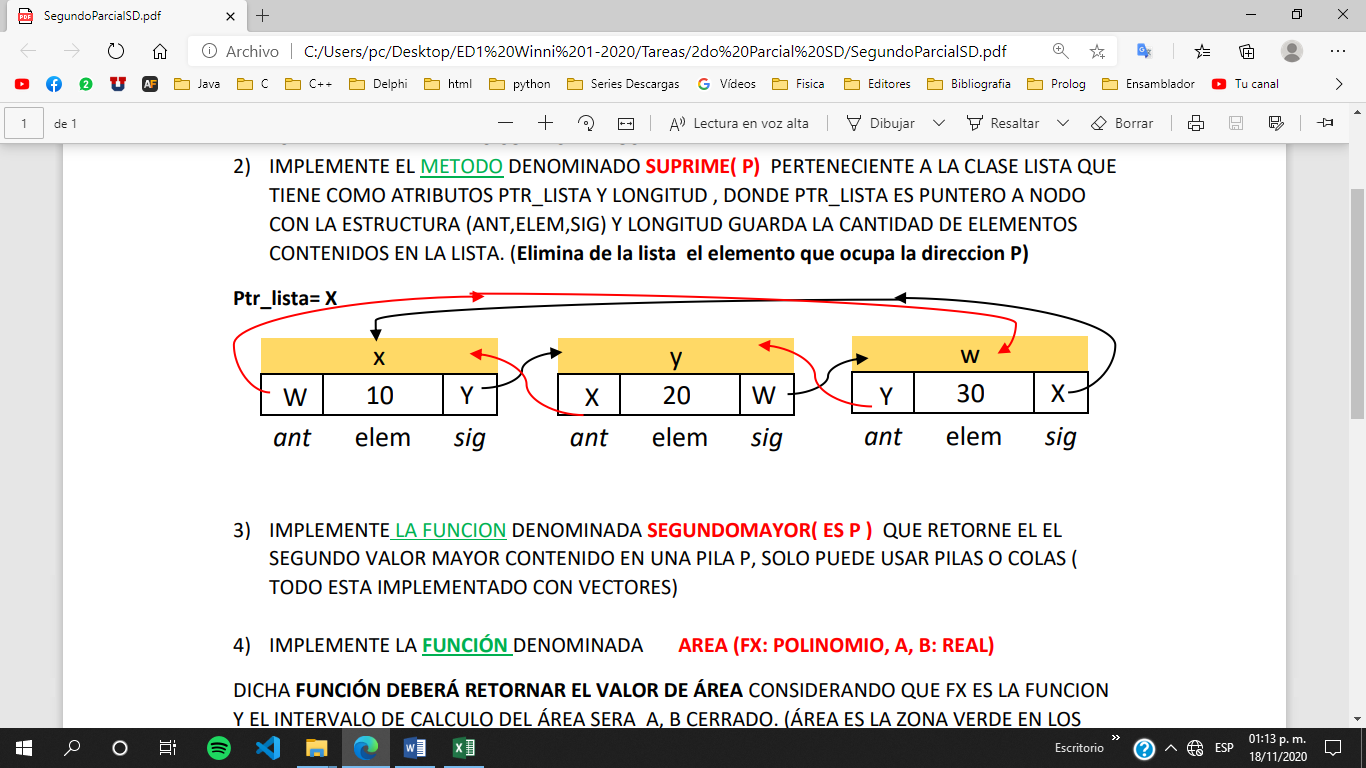
**void** inserta(Nodo\* P, **int** e);

**void** suprime(Nodo\* P);

**void** modifica(Nodo\* P, **int** e);

};

**Los enlaces y datos se almacenan de la siguiente forma:**



1. Implemente la función denominada: **int** segundo\_mayor(Pila &p);

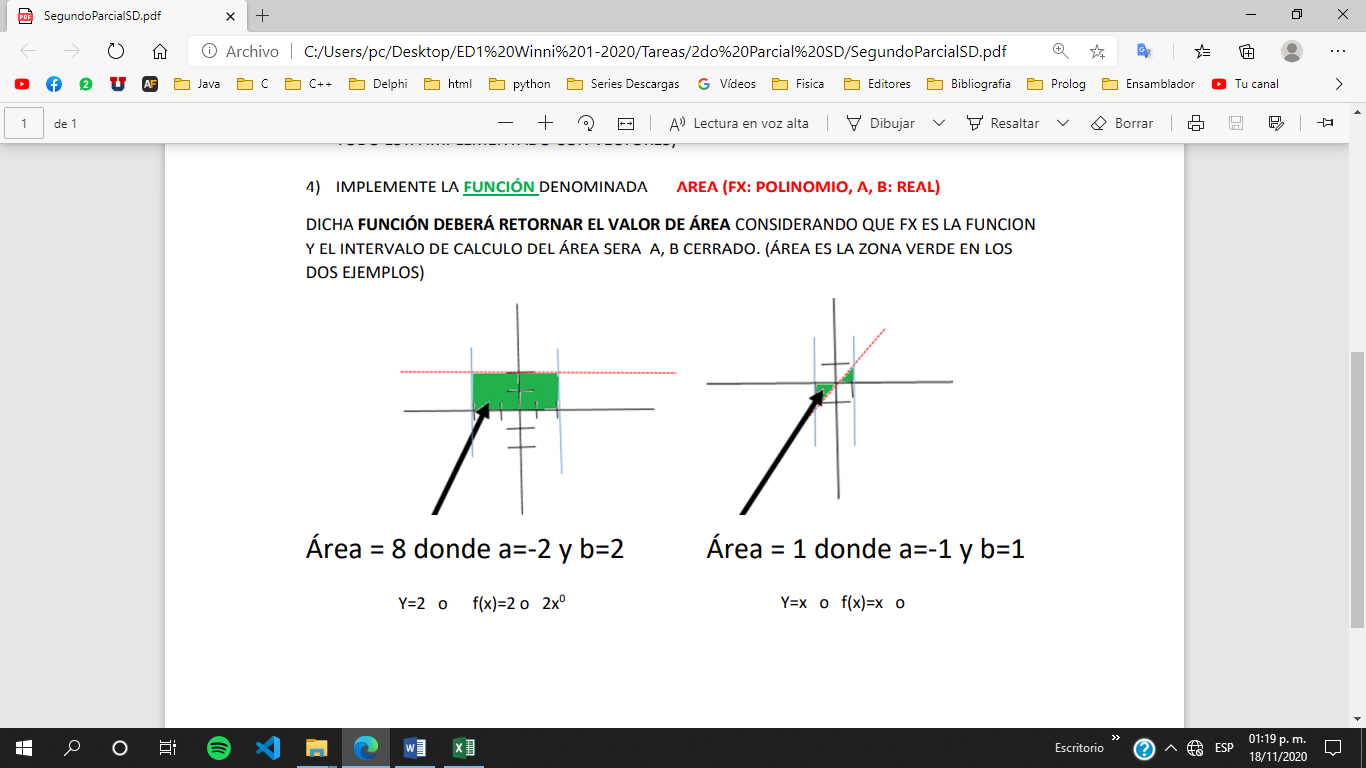
que retorne el **segundo mayor valor** contenido en una **Pila p**, solo puede usar Pilas o Colas de ayuda (Todo esta implementado con vectores)

1. Implemente la función denominada:

**float** area(Polinomio &fx, **float** a**, float** b);

considerando que **fx es la función** y el intervalo de cálculo del área será **a** y **b cerrado**.

**Ejm:**



**Interfaces de las clases programadas en ED1**

**Clase Memoria Métodos**

Memoria();//Constructor

direccionnew\_espacio(**string** ids);//direccion = int

**void** delete\_espacio(direccion dir);

**void** poner\_dato(direccion dir, **string** id, **int** valor);

**int** obtener\_dato(direccion dir, **int** lugar);

**int** espacio\_disponible();

**int** espacio\_ocupado();

**bool** direccion\_libre(direccion dir);

**Clase Lista Métodos**

/\* direccion puede ser un int o un puntero, depende

del tipo de implementación \*/

Lista();//Constructor

direccion fin();

direccion primero();

direccion siguiente(direccion dir);

direccion anterior(direccion dir);

**bool** vacia();

**int** recupera(direccion dir);

**int** longitud();

**void** inserta(direccion dir, **int** elemento);

**void** suprime(direccion dir);

**void** modifica(direccion dir, **int** elemento);

direccion localiza(**int** elemento);

**void** elimina\_dato(**int** elemento);

**void** anula();

**Clase Conjunto Métodos**

Conjunto();//Constructor

**bool** vacio();

**int** cardinal();

**int** ordinal(**int** elemento);

**void** inserta(**int** elemento);

**void** suprime(**int** elemento);

**bool** pertenece(**int** elemento);

**int** muestrea();

**Clase Polinomio Métodos**

Polinomio();//Constructor

**bool** es\_cero();

**void** poner\_termino(**int** coef, **int** exp);

**int** grado();

**int** coeficiente(**int** exponente);

**int** exponente(**int** nro\_termino);

**float** evaluar(**float** x);

**void** derivada();

**Clase MatrizDispersa Métodos**

MatrizDispersa();//

**void** dimensionar(**int** df, **int** dc);

**int** dimension\_fila();

**int** dimension\_columna();

**void** poner(**int** f, **int** c, **int** valor);

**int** elemento(**int** f, **int** c);

**void** definir\_valor\_repetido(**int** valor);

**Clase Pila Métodos**

Pila();//Constructor

**bool** vacia();

**void** meter(**int** elemento);

**void** sacar(**int** &elemento);

**int** cima();

**Clase Cola Métodos**

Cola();//Constructor

**bool** vacia();

**void** meter(**int** elemento);

**void** sacar(**int** &elemento);

**int** primero();

**Clase Dicola Métodos**

Dicola();//Constructor

**bool** vacia();

**void** meter(**int** elemento);

**void** sacar(**int** &elemento);

**void** meter\_frente(**int** elemento);

**void** sacar\_final(**int** &elemento);

**int** primero();

**Clase ColadePrioridad Métodos**

ColadePrioridad();//Constructor

**bool** vacia(); **int** primero();

**void** poner(**int** elemento, **int** prioridad);

**void** frecuencia\_prioridad(**int** frec, **int** prior);

**void** sacar(**int** &elemento);