I – RECURSIVIDAD

1. Mencione 5 ventajas de escribir programas recursivos.

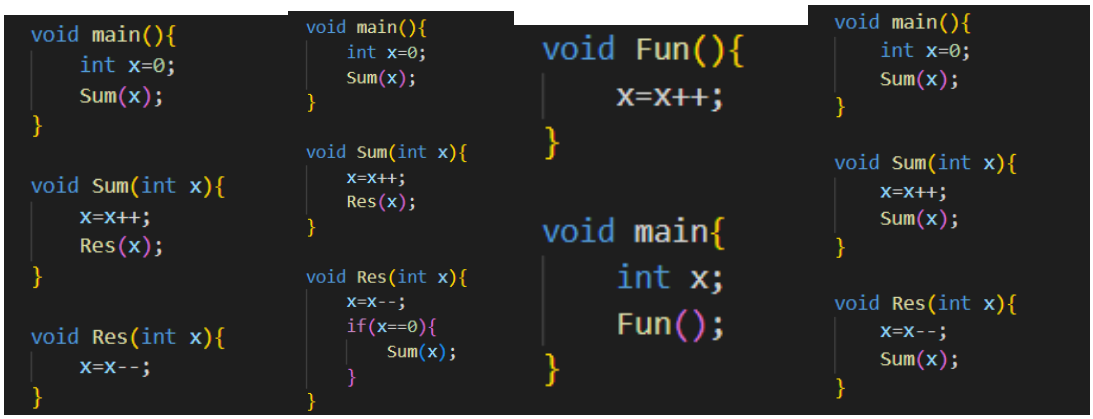
2. Si se manejan métodos con arreglos largos, ¿es conveniente usar recursividad? ¿Por qué?

3. ¿El siguiente código es recursivo? ¿Por qué?

```
void fun(int x) {

	for (int i = 1; i < x; i++) {
		sum = sum + (i * i);
	}

	cout << sum << endl;
}

int main() {

	cout << "Ingrese numero" << endl;
	cin >> num;

	fun(num);

}
```

4. Realice un programa que calcule el factorial de un número usando recursividad.

5. Identifique las funciones recursivas.



```
void main(){
    int x=0;
    Sum(x);
}

void Sum(int x){
    x=x++;
    Res(x);
}

void Res(int x){
    x=x--;
}
```

```
void main(){
    int x=0;
    Sum(x);
}

void Sum(int x){
    x=x++;
    Res(x);
}

void Res(int x){
    x=x--;
    if(x==0){
        Sum(x);
    }
}
```

```
void Fun(){
    x=x++;
}

void main{
    int x;
    Fun();
}
```

```
void main(){
    int x=0;
    Sum(x);
}

void Sum(int x){
    x=x++;
    Sum(x);
}

void Res(int x){
    x=x--;
    Sum(x);
}
```

6. Define la(s) diferencia(s) entre recursividad directa e indirecta.

7. ¿Cómo se detiene a las funciones recursivas?

II – PILAS Y COLAS

1. ¿Por qué la pila es una estructura de datos LIFO?

2. ¿Qué operación realiza este código?

```
void push(struct stack* S, int e) {
        S->top++;
        S->item[S->top] = e;
}
```

3. Identifica a la Pila y a la Cola

```
struct Nodo{
    int dato;
    Nodo *siguiente;
};
```

_____

```
struct Nodo2{
    int dato;
    Nodo2 *siguiente;
    Nodo2 *frente;
    Nodo2 *fin;
};
```

_____

4. En una escuela primaria recibieron varias cajas de libros, los cuales están siendo acomodados en pilas para entregar a los alumnos conforme vayan llegando. Realice un programa que permita insertar y eliminar datos de la pila de forma dinámica.

5. Identifica el tipo de compilación

   a. _____

   - **A + B** -> *A, +, B*
   - **(A + (B*C))** -> *A, +, B, C*

   b. _____

   - **3 + 4** -> *+, 3, 4*
   - **((A+B) + C)** -> *+, *, A, B, C*

   c. _____

   - **A+B** -> *A, B, +*
   - **((A+B) + C)** -> *A, B, C, +, **

6. Escriba las siguientes expresiones en notación postfija.
   a * (b * c) / d
   (2 + (3 * 4)) – x
   (-5 + 2) / (10*y – z)
   (b*a – w) + (x * (y/2))
   x-z * ((d*c) / a % y)
7. Invertir las expresiones postfijas anteriores a notación prefija.

8. En un consultorio médico se atiende a los pacientes conforme vayan entrando al consultorio. Realice un programa que registre a los pacientes en una estructura de tipo cola y los elimine al ser atendidos.

III – LISTAS LIGADAS SIMPLES

1. Describa brevemente las características del proceso de asignación de memoria en lenguajes como C y C + +.
2. Describa qué tipo de estructura de datos constituye una lista ligada.
3. Enliste 2 ventajas del uso de una lista ligada frente al uso de arreglos.
4. Enliste 2 desventajas del uso de una lista ligada frente al uso de arreglos.
5. Describa la estructura básica de un nodo en una lista ligada simple.
6. Aquí se describen 3 formas de acceder a la información utilizando los operadores de acceso "." y "->". Una de ellas no es correcta y ocasionaría un error. Indique cuál y explique por qué.
   - *lista->data;
   - (*lista).data;
   - *lista.data;
7. Escriba un código que le permita recorrer e imprimir la siguiente lista ligada simple, desde el inicio hasta el final.

```
Class StudentList(){
  struct StudentNode{
    char name[50];
    StudentNode* next;
  }
   StudentNode* start;
}
```

8. Realice un programa que permita gestionar una lista ligada simple de números enteros mediante un menú de opciones.

   Las opciones en dicho menú deberán permitir:

   a. Insertar un nuevo elemento al final de la lista.
   b. Eliminar un elemento de la lista tomando como parámetro su posición.
   c. Buscar un valor específico. Se deberá retornar la cantidad de coincidencias de dicho valor, ya sean cero o varias.
   d. Salir del programa.

IV – LISTAS LIGADAS DOBLES

1. Describa la diferencia entre una lista simple y una doblemente ligada.

2. Realice un programa que permita gestionar una lista doblemente ligada de números enteros mediante un menú de opciones.

   Las opciones en dicho menú deberán permitir:

   a. Insertar un nuevo elemento en la posición que se desee.
   b. Eliminar un elemento de la lista tomando como parámetro su posición.
   c. Ordenar la lista de forma ascendente (de menor a mayor) e imprimirla en pantalla
   d. Intercambiar el valor de dos elementos dadas sus posiciones.
   e. Salir del programa.

3. Realice un programa que permita ingresar números enteros en una lista doblemente ligada. La lista se deberá estar siempre visible y ordenada, imprimiendola después cada inserción

V – ÁRBOLES

1. Defina un árbol y qué tipo de estructura de datos constituye

2. ¿Los árboles son estructuras recursivas? Justifique.

3. Defina los siguientes conceptos relacionados con árboles
    a. Grado de un nodo:
    b. Raíz:
    c. Nodo intermedio:
    d. Hoja:
    e. Altura de un árbol:
    f. Grado de un árbol:
    g. Árbol sesgado:
4. Enliste las diferencias entre un árbol y un árbol binario

5. Defina los siguientes tipos de árboles
    a. Árbol binario

    b. Árbol hilvanado

    c. Árbol binario de búsqueda

    d. Árbol binario balanceado
6. Principal aplicación de los árboles binarios

7. Realice un programa que almacene los siguientes datos en un árbol binario de búsqueda (En ese orden).

- H, W, N, O, R, K, F, J, S, U, B, Z

Imprima el árbol siguiendo el recorrido en:

- Preorden
- Inorden
- Postorden

8. Realice un programa que permita gestionar, mediante un menú de opciones, la lista de los nombres de los empleados de una empresa. Dichos nombres deberán estar almacenados en un árbol binario de búsqueda. A su vez, cada empleado deberá contar un ID entero único para búsqueda. El menú deberá contar con las siguientes opciones:

    a. Ingresar un nuevo empleado

    b. Eliminar un empleado por id

    c. Buscar un empleado por id

    d. Salir del programa

VI – HEAP

1. Defina lo qué es una estructura heap

2. ¿Qué significa que un árbol está parcialmente ordenado?

3. ¿Cuál es la diferencia entre un árbol completo y uno incompleto?

4. Utilizando las propiedades de los árboles binarios, calcule la cantidad de nodos que tienen estos árboles completos mediante su altura.
    a. h = 2
    b. h = 4
    c. h = 7
    d. h = 10
5. Realice un programa que permita gestionar, mediante un menú de opciones, un árbol **heap mínimo** de números enteros, implementado en forma de arreglo. El menú deberá contar con las siguientes opciones:
    a. Insertar un nuevo número
    b. Eliminar un elemento tomando como parámetro su valor
    c. Extraer la raíz
    d. Salir del programa

VII – ORDENAMIENTO

Método Burbuja

1. Explique en qué consiste este método.

2. ¿Considera este, el mejor método de ordenamiento? Justifique su respuesta.

3. ¿Cuál será el valor de los elementos del vector después de tres pasadas más usando el algoritmo burbuja?

3	13	8	25	45	23	98	58

4. Realiza un algoritmo para ordenar el siguiente conjunto de elementos usando el método burbuja.

15	25	35	40	50	55	65	75

Método Quicksort

1. Este es considerado uno de los métodos más eficientes y adecuados para listas largas. ¿Podría explicar por qué es así?

2. ¿Cómo funciona el método Quicksort?

3. ¿Este algoritmo funciona? Justifique su respuesta.

```
void quicksort(double A[20], int primero, int ultimo)
{
    int central, i, j;
    double pivote;
    central = (primero + ultimo);
    pivote = A[central];
    i = primero;
    j = ultimo;
    do
    {
        while (A[i] < pivote) i++;
        while (A[j] > pivote) j--;
        if (i <= j)
        {
            double temp;
            temp = A[i];
            A[i] = A[j];
            A[j] = temp;
            i++;
            j--;
        }
    }

    while (i <= j);
    if (primero < j)
        quicksort(A, primero, j);
    if (i < ultimo)
        quicksort(A, i, ultimo);
}
```

4. Dada una array de números enteros positivos y negativos, separelos. La salida debe imprimir todos los números negativos, seguidos de todos los números positivos.

Heapsort

1. ¿Qué es el método heapsort?

2. ¿Cómo funciona el heapsort?

3. Nombre algunas de las ventajas del método heapsort.

4. Y, ¿qué desventajas conlleva usar este método?

5. Crea un algoritmo que ordene la cadena de números mostrada a continuación usando el método heapsort.

12 11 13 5 6 7

Método de inserción

1. ¿Podrías mencionar al menos dos ventajas del método de ordenamiento por inserción?

2. Existen dos tipos de ordenamiento por inserción. ¿Cuáles son?

3. ¿En qué situación sería más adecuado usar el método de inserción y por qué no otro como el quicksort?

4. ¿Cuál es la diferencia entre ordenación por selección y ordenación por inserción?

5. Diseña un algoritmo que ordene una lista de números con *n* elementos (que el usuario tenga que definir la cantidad de elementos de la lista), usando el método de ordenamiento por inserción directa.

Método de intercalación

1. ¿Qué similitudes encuentra con este método y el método Quicksort?

2. Sin embargo, estos dos algoritmos no son iguales, por lo que no hay que confundirlos. ¿Cuál sería la primera diferencia con la que podría identificarlo?

3. ¿Cuáles serían las ventajas y desventajas de usar este método?

4. Cuando los datos están almacenados en vectores, tablas (arrays), lista enlazadas o árboles, la ordenación se denomina _____. Cuando los datos a clasificar se encuentran almacenados en archivos, en soporte de almacenamiento masivo (cintas o discos) el proceso de ordenación se denomina _____.

5. Diseña un algoritmo que ordene una lista de elementos usando el método de intercalación con una cadena de n números.

VIII - BÚSQUEDA

Secuencial

1. Existen dos tipos de búsqueda. ¿Cuáles son?
2. ¿En dónde se centra la búsqueda interna?
3. ¿En dónde se guardan los elementos en una búsqueda externa?
4. ¿Qué es lo que se necesita para llevar a cabo la búsqueda secuencial?
5. ¿Cuáles serían las principales ventajas de usar este método de búsqueda?
6. ¿Cuál cree que sería el mayor defecto?
7. Mediante programación en C + +, realice un programa que busque un elemento mediante una búsqueda secuencial.

Búsqueda binaria

1. ¿En qué consiste la búsqueda binaria?
2. ¿Cuál es el mejor caso que podría tener la búsqueda binaria?
3. ¿Y cuál podría ser el peor caso?
4. ¿Por qué es obligatorio que el arreglo esté ordenado para este método?
5. Realice un programa que capture una cadena de 12 números y que contenga el número 5 y busque ese número (use cualquier método para ordenar la cadena del array.)

IX - GRAFOS

1. ¿Qué es un grafo?
2. ¿Qué usos se les puede dar a los grafos?
3. Define:
    - Vértice:
    - Arco:
    - Cabeza:
    - Cola:
    - Grado de un vértice:
    - Grafos direccionados:
4. ¿Qué tipos de grafos conoce?
5. ¿Cuáles son los multigrafos?
6. ¿Qué son los hipergrafos?