1. Write a well-commented example code for storing student details (also functions for performing CRUD) using pointers and structures in C and similarly on classes and objects in C++, Java, or Python.

The data structure used to store the records is linked list.

The insertion, update, deletion time is O(n).

In C the student record is maintained in the structure Student.

Whereas in Python the data record is maintained in Json format.

C is a procedure-oriented language so we have to make different functions for CRUD operation and we have to take care of data types and memory management to avoid any memory leak.

Python is an Object-oriented language (in python everything is an object) thus everything is encapsulated inside a class and we don't have to take care of the data types, memory management done automatically by garbage collector.

From user point of view all the CRUD operation is fairly simple in both the language.

In C we directly have to call the c,r,u,d function from the main program.

In Python first we need to initiate an object and through this object we can call the various methods.

2. Create a linked list in C (Using structures and Pointers) and also create a linked list in C++, JAVA, Python (Using classes and objects).

The basic structure of linked list comprises of data and next pointer field.

To insert at the end of a linked list we require to traverse untill the last node and hence it takes O(n) time.

If we maintain a tail pointer than the insertion time will reduce to O(1).

To delete or update an element in a linked list we require O(n) time anyway.

In C linked list is implemented using structure and pointers.

The basic structure of a node in a linked list in C is something like

```
struct node {
   stud data;
   struct node *next;
};
```

In python it is implemented using Class, whenever an object is initiated the __init__ method will invoke.

Hence, we can do initial setup for any data structure through __init__ which I have used throughout in all the data structures.

I have tried to maintain the similarity between codes written in C as well as in Python.

Hence all the methods and its working is almost similar in both the language, for e.g a function insertnode is used to insert a new node in the linked list, it is implemented quite similarly both in Python and C.

3. Create a doubly linked list C (Using structures and Pointers) and also create a doubly linked list in C++, JAVA, Python (Using classes and objects).

A doubly linked list is a linear linked list which can be traversed both forward and backward from any node.

For this we have to maintain two pointers "Next" and "Previous".

The basic structure of DLL in C is

```
struct node{
   int data;
   struct node *next;
   struct node *previous;
};
```

Similarly in python is

```
class Node:
   def __init__(self,data):
      self.data=data
      self.next=None
      self.previous=None
```

The function insertnode will insert the node at the end of the list.

Traverse back and forth method is used first to traverse from start and reach to the end then from the end again reach to the start.

A function delete list is written in C to delete the entire list whereas it is not required in python as Python's memory management will take care of the unused space once the program ends.

4. Create a circular linked list C (Using structures and Pointers) and also create a circular linked list in C++, JAVA,Python (Using classes and objects).

A circular linked list is a variation of linear linked list in which the last node points to the first node (instead of null) to maintain a circular structure. Application of Circular LL is in the implementation of Queue with one pointer with both enqueue and dequeue operation in O(1) time.

The basic structure of circular linked list is similar to linear linked list.

I have implemented circular linked list using a linear linked list and two pointers (head and tail).

The insertnode function will insert at the next of tail and hence require only O(1) time.

There is slight variation in traverselist function which first visits from head to tail and then moves back to head.

5. Create a Stack C (Using structures and Pointers) and also create a Stack in C++, JAVA, Python (Using classes and objects).

Stack is a linear data structure which follows the Last in first out (LIFO) order.

Application of stack is in infix-post conversion,dfs,recursion,compilers etc.

I have implemented stack using linear linked list, the element is added in the front of the linked list hence every push operation takes O(1) time. Head pointer will act as the top of stack pointers using this pop operation is implemented in O(1) time.

As linked list is a dynamic data structure we dont need to check the overflow condition but we need to check the underflow condition to avoid segmentation fault and dangling pointers.

6. Create a Queue C (Using structures and Pointers) and also create a Queue in C++, JAVA, Python (Using classes and objects).

Queue is a linear data structure which follows the First in First out(FIFO) order.

Application of queue is in various scheduling algorithm,bfs etc.

I have implemented Queue using linear linked list with two pointers(head and tail) so that both enqueue and dequeue operation will perform in O(1) time.

The enqueue operation basically insert the node next to the tail, the dequeue operation will extract the node from the front.

7. Create a Deque C (Using structures and Pointers) and also create a Deque in C++, JAVA, Python (Using classes and objects).

Double Ended Queue knowns as Deque is a variation of Queue in which both insertion and deletion takes place from both the end.

There are various operation in a deque namely:

enqueue_front(): Enqueue at front of the node, for e.g if we have deque with elements [1,2,3] after enqueue_front(4) deque will become [4,1,2,3]

enqueue_rear(): Enqueue at end of the node, for e.g if we have deque with elements [1,2,3] after enqueue_rear(4) deque will become [1,2,3,4]

dequeue_front(): Dequeue from the front. for e.g [1 2 3 4] after dequeue_front will be come [2,3,4]

dequeue_rear(): Dequeue from the rear. for e.g [1 2 3 4] after dequeue_rear will become [1,2,3]

I have implemented deque using doubly linked list and two pointers so that all the operation can be perform in O(1) time.

Enqueue_front the node is inserted in begining using head pointer.

Enqueue_rear the node is inserted at the end using tail pointer.

8. Create a Circular Queue C (Using structures and Pointers) and also create a Circular Queue in C++, JAVA, Python (Using classes and objects).

Circular Queue is the implementation of queue using Circular Array.

Circular array is linear array which is viewed as circular with the help of modular arithmetic.

I have implemented circular queue with array and two pointers front and rear.

Overflow condition=> when front==(rear+1) mod size

Underflow condition=> when front==rear

9. Create a Priority Queue C (Using structures and Pointers) and also create a Priority Queue in C++, JAVA,Python (Using classes and objects).

A priority queue is a variation of queue with priority associated with every element.

An element with highest priority is dequeued first.

If two elements have same priority then dequeued in the order how they enqueued.

The basic structure of Priority Queue is

```
struct node{
   int data;
   int priority;
   struct node *next;
};
```

The enqueue operation is similar to that of normal Queue.

For dequeue I have used peek() function which will lookup the entire list and return the highest priority. It takes O(n) time.

The dequeue() function will call this peek function and than traverse the list again to match with the highest priority, hence time complexity for n dequeue operation will be O(n^2).

10. Create a Binary search tree C (Using structures and Pointers) and also create a Binary search tree in C++, JAVA, Python (Using classes and objects)

Binary Search Tree is a nonlinear data structure. It is a rooted binary tree with the following rules:

Element < Parent will be stored on left and >= Parent will store on right.

The inorder traversal of binary search tree will extract the elements in sorted order.

The worst-case time complexity to construct a bst is O(n^2) in case when the tree is skewed.

The worst-case search time of any element is O(n).

The basic structure of a node in bst is

```
struct node{
   int data;
   struct node *left;
   struct node *right;
   };
```