Assignment-40: Array

1. Define a class Array to implement array data structure with member variables to store capacity of array, last index of the last filled block of the array and a pointer to hold the address of the first block of the dynamically created array.
2. In question 1, define a parameterised constructor to create an array of specified size.
3. In the question 1, add a method to check whether an array is empty or not by returning True or False.
4. In question 1, define a method to append a new element in the array
5. In question 1, define a method to insert a new element at specified index
6. In question 1, define a method to edit an element at specified index.
7. In question 1, define a method to delete an element at specified index.
8. In question 1, define a method to check if the array is full by returning true or false.
9. In question 1, define a method to get element at specified index.
10. In question 1, define a method to count number of elements present in the array.
11. In question 1, define a destructor to deallocate the memory of array.
12. In question 1, define a method to find an element in the array. Return index if the element found, otherwise return -1.

Assignment-41: Problems on Arrays

1. Define a function to sort elements of the array.
2. Define a function to find the greatest element in the array
3. Define a function to find the smallest element in the array
4. Define a function to search an element in the array
5. Define a function to calculate sum of all the elements of an array.
6. Define a function to calculate average of all the elements of an array.
7. Define a function to rotate an array towards right by one position
8. Define a function to remove duplicate elements in the array.
9. Define a function to find the second largest element in the array.
10. Define a function to swap elements with specified indices in the array.

Assignment-42: Dynamic Arrays

1. Define a class DynArray to implement dynamic array data structure with member variables to store capacity of array, last index of the last filled block of the array and a pointer to hold the address of the first block of the dynamically created array.
2. In question 1, define a parameterised constructor to create an array of specified size.
3. In question 1, define a method doubleArray() to increase the size of the array by double of its size.
4. In question 1, define a method halfArray() to decrease the size of the array by half of its size.
5. In question 1, define a method which returns the current capacity of the array.
6. In the question 1, add a method to check whether an array is empty or not by returning True or False.
7. In question 1, define a method to append a new element in the array
8. In question 1, define a method to insert a new element at specified index
9. In question 1, define a method to edit an element at specified index.
10. In question 1, define a method to delete an element at specified index.
11. In question 1, define a method to check if the array is full by returning true or false.
12. In question 1, define a method to get element at specified index.
13. In question 1, define a method to count number of elements present in the array.
14. In question 1, define a destructor to deallocate the memory of array.
15. In question 1, define a method to find an element in the array. Return index if the element found, otherwise return -1.
16. In question 1, define a copy constructor to implement deep copy.
17. In question 1, define a copy assignment operator to implement deep copy.

Assignment-43: Singly Linked List

1. Define a class SLL to implement singly linked list data structure with member variable start pointer of type node.
2. In question 1, define a constructor to initialise start pointer with NULL.
3. In question 1, define a method to insert a data into the list at the beginning.
4. In question 1, define a method to insert a data into the list at the end
5. In question 1, define a method to search a node with the give item.
6. In question 1, define a method to insert a data into the list after the specified node of the list.
7. In question 1, define a method to delete first node from the list.
8. In question 1, define a method to delete last node of the list.
9. In question 1, define a method to delete a specific node.
10. In question 1, define a destructor to deallocates memory for all the nodes in the linked list.
11. In question 1, define a copy constructor to implement deep copy.
12. In question 1, define a copy assignment operator to implement deep copy.

Assignment-44: Doubly Linked List

1. Define a class DLL to implement singly linked list data structure with member variable start pointer of type node.
2. In question 1, define a constructor to initialise start pointer with NULL.
3. In question 1, define a method to insert a data into the list at the beginning.
4. In question 1, define a method to insert a data into the list at the end
5. In question 1, define a method to search a node with the give item.
6. In question 1, define a method to insert a data into the list after the specified node of the list.
7. In question 1, define a method to delete first node from the list.
8. In question 1, define a method to delete last node of the list.
9. In question 1, define a method to delete a specific node.
10. In question 1, define a destructor to deallocates memory for all the nodes in the linked list.
11. In question 1, define a copy constructor to implement deep copy.
12. In question 1, define a copy assignment operator to implement deep copy.

Assignment-45: Circular Linked List

1. Define a class CLL to implement Circular linked list data structure with member variable last pointer of type node.
2. In question 1, define a constructor to initialise last pointer with NULL.
3. In question 1, define a method to insert a data into the list at the beginning.
4. In question 1, define a method to insert a data into the list at the end
5. In question 1, define a method to search a node with the give item.
6. In question 1, define a method to insert a data into the list after the specified node of the list.
7. In question 1, define a method to delete first node from the list.
8. In question 1, define a method to delete last node of the list.
9. In question 1, define a method to delete a specific node.
10. In question 1, define a destructor to deallocates memory for all the nodes in the linked list.
11. In question 1, define a copy constructor to implement deep copy.
12. In question 1, define a copy assignment operator to implement deep copy.

Assignment-46: Circular Doubly Linked List

1. Define a class CDLL to implement Circular Doubly linked list data structure with member variable start pointer of type node.
2. In question 1, define a constructor to initialise start pointer with NULL.
3. In question 1, define a method to insert a data into the list at the beginning.
4. In question 1, define a method to insert a data into the list at the end
5. In question 1, define a method to search a node with the give item.
6. In question 1, define a method to insert a data into the list after the specified node of the list.
7. In question 1, define a method to delete first node from the list.
8. In question 1, define a method to delete last node of the list.
9. In question 1, define a method to delete a specific node.
10. In question 1, define a destructor to deallocates memory for all the nodes in the linked list.
11. In question 1, define a copy constructor to implement deep copy.
12. In question 1, define a copy assignment operator to implement deep copy.

Assignment-47: Stack using arrays

1. Define a class Stack with capacity, top and ptr pointer as member variables. Implement stack using array.
2. In question 1, define a parameterzied constructor to initialise member variables.
3. In question 1, define a method to push a new element on to the Stack.
4. In question 1, define a method to peek top element of the stack.
5. In question 1, define a method to pop the top element of the stack.
6. In question 1, define a destructor to deallocates the memory.
7. In question 1, define a method to check stack overflow
8. In question 1, define a method to check stack underflow.
9. In question 1, define a copy constructor to implement deep copy.
10. In question 1, define a copy assignment operator to implement deep copy.

Assignment-48: Stack using linked list

1. Define a class Stack with node type pointer **top** as member variable. Implement stack using linked list.
2. In question 1, define a constructor to initialise member variable.
3. In question 1, define a method to push a new element on to the Stack.
4. In question 1, define a method to peek top element of the stack.
5. In question 1, define a method to pop the top element of the stack.
6. In question 1, define a destructor to deallocates the memory.
7. In question 1, define a copy constructor to implement deep copy.
8. In question 1, define a copy assignment operator to implement deep copy.
9. Define a method to reverse a stack.

Assignment-49: Queue using arrays

1. Define a class Queue with capacity, front, rear and ptr pointer as member variables. Implement queue using array.
2. In question 1, define a parameterzied constructor to initialise member variables.
3. In question 1, define a method to insert a new element at the rear in the queue.
4. In question 1, define a method to view rear element of the queue.
5. In question 1, define a method to view front element of the queue.
6. In question 1, define a method to delete the front element of the queue.
7. In question 1, define a destructor to deallocates the memory.
8. In question 1, define a method to check queue overflow
9. In question 1, define a method to check queue underflow.
10. In question 1, Define a method to count number of elements present in the queue
11. In question 1, define a copy constructor to implement deep copy.
12. In question 1, define a copy assignment operator to implement deep copy.

Assignment-50: Queue using linked list

1. Define a class Queue with node type pointers front and rear as member variables. Implement queue using Singly linked list.
2. In question 1, define a constructor to initialise member variable.
3. In question 1, define a method to insert a new element at the rear in the queue.
4. In question 1, define a method to view rear element in the queue.
5. In question 1, define a method to view front element in the queue.
6. In question 1, define a method to delete the front element of the queue.
7. In question 1, define a destructor to deallocates the memory.
8. In question 1, define a copy constructor to implement deep copy.
9. In question 1, define a copy assignment operator to implement deep copy.
10. In question 1, define a method to count number of elements present in the queue.

Assignment-51: Deque

1. Define a class Deque with node type pointers front and rear as member variables. Implement queue using doubly linked list.
2. In question 1, define a constructor to initialise member variables
3. In question 1, define a method to insert a new element at the front
4. In question 1, define a method to insert a new element at the rear.
5. In question 1, define a method to delete front element
6. In question 1, define a method to delete rear element
7. In question 1, define a method to get front element.
8. In question 1, define a method to get rear element
9. In question 1, define a destructor to deallocate the memory.
10. In question 1, define a method to check if deque is empty.
11. In question 1, define a copy constructor to implement deep copy.
12. In question 1, define a copy assignment operator to implement deep copy.

Assignment-52: Priority Queue

1. Define a class PriorityQueue with node type pointer start as member variable. Implement PriorityQueue using singly linked list.
2. In question 1, define a constructor to initialise member variable.
3. In question 1, define a method to insert new item in the Priority Queue according to the priority number.
4. In question 1, define a method to delete highest priority element
5. In question 1, define a method to get highest priority element
6. In question 1, define a method to get highest priority number.
7. In question 1, define a destructor to deallocate the memory.
8. In question 1, define a method to check if Priority Queue is empty
9. In question 1, define a copy constructor to implement deep copy.
10. In question 1, define a copy assignment operator to implement deep copy.

Assignment-53: Tree

1. Define a class BST (Binary Search Tree) with node type pointer root as member variable. Implement Binary Search Tree using linked representation.
2. In question 1, define a constructor to initialise root pointer with NULL.
3. In question 1, define a method to check if the tree is empty.
4. In question 1, define a method to insert a new element in the BST
5. In question 1, define a method for preorder traversing of BST
6. In question 1, define a method for inorder traversing of BST
7. In question 1, define a method for postorder traversing of BST
8. In question 1, define a method to delete an element from BST
9. In question 1, define a method to search an item in the BST
10. In question 1, define a destructor to release memory of all the nodes of BST.
11. In question 1, define a copy constructor to implement deep copy.
12. In question 1, define a copy assignment operator to implement deep copy.

Assignment-54: AVL Tree

1. Define a class AVL, with node type pointer root as member variable. Implement AVL tree using linked representation
2. In question 1, define constructor to initialise member variable.
3. In question 1, define a method to get balance factor of a node.
4. In question 1, define a method leftRotate for anticlockwise rotation
5. In question 1, define a method rightRotate for clockwise rotation
6. In question 1, define a method to insert new element in the tree
7. In question 1, define a method to delete an element from the tree.
8. In question 1, define preorder traversal
9. In question 1, define postorder traversal
10. In question 1, define inorder traversal.
11. In question 1, define destructor to release memory.
12. In question 1, define a search method to find the element in the tree.

Assignment-55: Graph Matrix

1. Define a class Graph using matrix representation with v_count, e_count and adj pointer as instance members.
2. In Question 1, define a method createGraph() to create and store adjacent node information.
3. In question 1, define a method to print graph matrix.
4. In question 1, define a method to print all the adjacent nodes of a given node.
5. In question 1, define a method to check if a given node is isolated node.
6. In question 1, define a destructor to deallocates memory

Assignment-56: Graph List Representation

1. Define a class Graph to implement linked list representation of graph. Define needful structure for node and class for AdjList.
2. Define appropriate constructors in the classes AdjList and Graph
3. Define appropriate methods to manage linked list in AdjList
4. Define createGraph() method in Graph class to allocate memory for array of AdjList Objects.
5. Define a method addEdge() in Graph class to add a new node in adjacency list.
6. Define destructors in the classes AdjList and Graph
7. Define a method to print graph (print values of adjacency list).

Assignment-57: Graph Traversing

1. Implement BFS in Graph Matrix.
2. Implement DFS in Graph Matrix.
3. Implement BFS in Graph list representation
4. Implement DFS in Graph list representation

Assignment-58: Sorting

1. Define a function to implement bubble sort
2. Define a function to implement modified bubble sort to achieve O(n) time complexity in best case.
3. Define a function to implement insertion sort.
4. Define a function to implement selection sort.
5. Define a function to implement quick sort
6. Define a function to implement merge sort
7. Define a class Employee with emp_id, name, salary as instance variables. Provide setters and getters in the class to access instance variables. Also define a function to sort Employee Array data by salary. Use Merge Sort.
8. In question-7, define a function to sort Employee Array data by name. Use Quick Sort.

Assignment-59: Heap, Searching

1. Define a function to implement linear search.
2. Define a function to implement binary search.
3. Define a class Heap with member variables capacity, lastIndex and pointer ptr.
4. In question 3, define a parameterised constructor to create a heap of given size.
5. In question 3, define a function to insert data in the heap.
6. In question 3, define a function to get the top element of the heap.
7. In question 3, define a function to delete a data from the heap.
8. Define a function to implement heap sort.

Assignment-60: Hashing

1. To store student records use hash table data structure. Define a class Student with member variables rollNumber, name, stream, year. Also define appropriate member function to handle Student class.
2. Define a structure node with member variables item of type Student and next pointer.
3. Define a class HashTable with member variables capacity, a node pointer to point an array of node pointers (refer Question 2). Implement hashing with the help of open hashing method.
4. Define a hashFunction which takes roll number as a key and return index number of the array of node pointers (refer Question 3).
5. Define a function to insert Student data in the HashTable
6. Define a search function to find the Student data using rollNumber.

Assignment-61: Template

1. Define a function template which takes two arguments of same types and return the greater value.
2. Define a function template to print values of an array of any type.
3. Define a function template to sort an array of any type.
4. Define data structure Array using class template
5. Define data structure Dynamic Array using class template
6. Define data structure Linked List using class template
7. Define data structure Doubly Linked List using class template
8. Define data structure Stack using class template
9. Define data structure Queue using class template
10. Define data structure deque using class template