

PRACTICALS

SUBJECT : DATA STRUCTURE (COCSC202)

Q1. Write a program to insert one element in an array and delete an element from an array.

Write a program to search for a number in an array.

Code:

```
1  #include<bits/stdc++.h>
2  using namespace std;
3
4  void insertElement(int arr[], int &s, int element, int posn) {
5      if (posn < 0 || posn > s) {
6          cout << "Invalid position!\n";
7          return;
8      }
9      for (int i = s; i > posn; i--) {
10         arr[i] = arr[i - 1];
11     }
12
13     arr[posn] = element;
14     s++;
15 }
16 void deleteElement(int arr[], int &size, int posn) {
17     if (posn < 0 || posn ≥ size) {
18         cout << "invalid \n";
19         return;
20     }
21     for (int i = posn; i < size - 1; i++)arr[i] = arr[i + 1];
22     size--;
23 }
24
25 int searchElement(int arr[], int size, int element) {
26     for (int i = 0; i < size; i++) {
27         if (arr[i] == element)return i;
28     }
29     return -1;
30 }
31
32 void displayArray(int arr[], int size) {
33     for (int i = 0; i < size; i++)cout << arr[i] << " ";
34     cout << "\n";
35 }
36
37 int main() {
38     int arr[100] = {1, 2, 3, 4, 5}; // Initial array
39     int size = 5;
40
41     cout << "orig Arr: ";
42     displayArray(arr, size);
43
44     insertElement(arr, size, 10, 2);
45     cout << "new Arr: ";
46     displayArray(arr, size);
47
48     deleteElement(arr, size, 3);
49     cout << "After Deletion: ";
50     displayArray(arr, size);
51
52     int element = 10;
53     int index = searchElement(arr, size, element);
54     if (index ≠ -1)
55         cout << index << "\n";
56     else
57         cout << " not found!" << "\n";
58     return 0;
59 }
```

Output:

```
Microsoft Windows [Version 10.0.26100.3476]
(c) Microsoft Corporation. All rights reserved.

D:\sem 2 dsa>cd "d:\sem 2 dsa\" && g++ arrays.cpp -o arrays && "d:\sem 2 dsa\"arrays
orig Arr: 1 2 3 4 5
new Arr: 1 2 10 3 4 5
After Deletion: 1 2 10 4 5
2
```

Topic : Stacks

Q2. Write a program to implement a various operations of stack using static and binary data structure.

Code:

```
1  #include <bits/stdc++.h>
2  using namespace std;
3
4  #define MAX 5
5
6  class Stack {
7      int top;
8      int arr[MAX];
9
10 public:
11     Stack() { top = -1; }
12     void push(int value) {
13         if (top == MAX - 1) {
14             cout << "Stack Overflow! " << "\n";
15             return;
16         }
17         arr[++top] = value;
18         cout << "Pushed: " << value << endl;
19     }
20     void pop() {
21         if (top == -1) {
22             cout << "Stack Underflow." << "\n";
23             return;
24         }
25         cout << "Popped: " << arr[top--] << "\n";
26     }
```

```
27     void peek() {
28         if (top == -1) {
29             cout << "empty" << "\n";
30             return;
31         }
32         cout << "Top: " << arr[top] << "\n";
33     }
34     void display() {
35         if (top == -1) {
36             cout << "Stack is empty!" << "\n";
37             return;
38         }
39         cout << "Stack elements: ";
40         for (int i = 0; i <= top; i++) cout << arr[i] << " ";
41         cout << "\n";
42     }
43 };
44
45 int main() {
46     Stack s;
47     s.push(10);
48     s.push(20);
49     s.push(30);
50     s.display();
```

```
    s.pop();
    s.peek();
    s.display();
    return 0;
}
```

Dynamic Representation Using Linked Lists

```
4 struct Node {
5     int data;
6     Node* next;
7 };
8
9 class Stack {
10     Node* top;
11
12 public:
13     Stack() { top = NULL; }
14     void push(int value) {
15         Node* newNode = new Node();
16         newNode->data = value;
17         newNode->next = top;
18         top = newNode;
19         cout << "Pushed: " << value << endl;
20     }
21     void pop() {
22         if (top == NULL) {
23             cout << "Stack Underflow!" << endl;
24             return;
25         }
26         Node* temp = top;
27         cout << "Popped: " << top->data << endl;
28         top = top->next;
29         delete temp;
```

```
31     void peek() {
32         if (top == NULL) {
33             cout << "empty!" << endl;
34             return;
35         }
36         cout << "Top element: " << top->data << endl;
37     }
38     void display() {
39         if (top == NULL) {
40             cout << "Stack is empty!" << endl;
41             return;
42         }
43         Node* temp = top;
44         cout << "elements: ";
45         while (temp != NULL) {
46             cout << temp->data << " ";
47             temp = temp->next;
48         }
49         cout << endl;
50     }
51 };
```

Output:

```
Microsoft Windows [Version 10.0.26100.3476]
(c) Microsoft Corporation. All rights reserved.

D:\sem 2 dsa>cd "d:\sem 2 dsa\" && g++ stkOperations.cpp -o stkOperations && "d:\sem 2 dsa\"stkOperations
Pushed: 10
Pushed: 20
Pushed: 30
Stack elements: 10 20 30
Popped: 30
Top: 20
Stack elements: 10 20

d:\sem 2 dsa>
```

```
D:\sem 2 dsa>cd "d:\sem 2 dsa\" && g++ stkDynamic.cpp -o stkDynamic && "d:\sem 2 dsa\"stkDynamic
Pushed: 10
Pushed: 20
Pushed: 30
elements: 30 20 10
Popped: 30
Top element: 20
elements: 20 10

d:\sem 2 dsa>
```

Q3. Write a program to implement a various operations of queue using static and binary data structure.

Code:

```
1  #include<bits/stdc++.h>
2  using namespace std;
3
4  #define MAX 5
5  class Queue {
6      int arr[MAX];
7      int fr, re;
8  public:
9      Queue() {
10         fr = -1;
11         re = -1;
12     }
13     void enqueue(int value) {
14         if (re == MAX - 1) {
15             cout << "Queue Overflow" << value << endl;
16             return;
17         }
18         if (fr == -1) fr = 0;
19         arr[++re] = value;
20         cout << "Enqueued: " << value << endl;
21     }
22     void dequeue() {
23         if (fr == -1 || fr > re) {
24             cout << "Queue Underflow" << endl;
25             return;
26         }
27         cout << "Dequeued: " << arr[fr++] << endl;
28         if (fr > re) fr = re = -1;
29     }
30     void peek() {
31         if (fr == -1) {
32             cout << "empty!" << endl;
33             return;
34         }
35         cout << "Front element: " << arr[fr] << endl;
36     }
37     void display() {
38         if (fr == -1) {
39             cout << "Queue is empty!" << endl;
40             return;
41         }
42         cout << "Queue elements: ";
43         for (int i = fr; i <= re; i++)
44             cout << arr[i] << " ";
45         cout << endl;
46     }
47 };
48 int main() {
49     Queue q;
50     q.enqueue(10);
51     q.enqueue(20);
52     q.enqueue(30);
53     q.display();
54     q.dequeue();
55     q.peek();
56     q.display();
```

Dynamic Representation

```
50 struct Node {
51     int data;
52     Node* next;
53 };
54 class Queue {
55     Node *front, *rear;
56 public:
57     Queue() {
58         front = rear = NULL;
59     }
60     void enqueue(int value) {
61         Node* newNode = new Node();
62         newNode->data = value;
63         newNode->next = NULL;
64
65         if (rear == NULL) {
66             front = rear = newNode;
67         } else {
68             rear->next = newNode;
69             rear = newNode;
70         }
71         cout << "Enqueued: " << value << endl;
72     }
73     void dequeue() {
74         if (front == NULL) {
75             cout << "Queue Underflow" << endl;
76             return;
77         }
78         Node* temp = front;
```

```
79         front = front->next;
80         cout << "Dequeued: " << temp->data << endl;
81         delete temp;
82
83         if (front == NULL) rear = NULL;
84     }
85     void peek() {
86         if (front == NULL) {
87             cout << "empty" << endl;
88             return;
89         }
90         cout << "Front element:" << front->data << endl;
91     }
92     void display() {
93         if (front == NULL) {
94             cout << "empty!" << endl;
95             return;
96         }
97         Node* temp = front;
98         cout << " ";
99         while (temp != NULL) {
100             cout << temp->data << " ";
101             temp = temp->next;
102         }
103         cout << endl;
104     }
105 };
```

Output:

```
Microsoft Windows [Version 10.0.26100.3476]
(c) Microsoft Corporation. All rights reserved.

D:\sem 2 dsa>cd "d:\sem 2 dsa\" && g++ queueOperations.cpp -o queueOperations && "d:\sem 2 dsa\"queueOperations
Enqueued: 10
Enqueued: 20
Enqueued: 30
Queue elements: 10 20 30
Dequeued: 10
Front element: 20
Queue elements: 20 30
```

Linked Lists

Q4. Write a program to implement a linked list i.e., singly linked list, doubly linked list. Write a program to insert a node in a linked list and delete a node from a linked list

Code:

```
3 struct Node {
4     int data;
5     Node* next;
6 };
7
8 class SinglyLL {
9     Node* head;
10
11 public:
12     SinglyLL() { head = NULL; }
13     void insert(int value) {
14         Node* newNode = new Node();
15         newNode->data = value;
16         newNode->next = NULL;
17
18         if (head == NULL) {
19             head = newNode;
20         } else {
21             Node* temp = head;
22             while (temp->next != NULL)
23                 temp = temp->next;
24             temp->next = newNode;
25         }
26         cout << "Inserted: " << value << endl;
27     }
28     void remove(int value) {
29         if (head == NULL) {
```

```
30             cout << "empty!" << endl;
31             return;
32         }
33         if (head->data == value) {
34             Node* temp = head;
35             head = head->next;
36             delete temp;
37             cout << value << endl;
38             return;
39         }
40         Node* temp = head;
41         while (temp->next != NULL && temp->next->data != value)
42             temp = temp->next;
43         if (temp->next == NULL) {
44             cout << "Not Found" << endl;
45             return;
46         }
47         Node* toDelete = temp->next;
48         temp->next = toDelete->next;
49         delete toDelete;
50         cout << "Deleted: " << value << endl;
51     }
52     void display() {
53         Node* temp = head;
54         if (!temp) {
55             cout << "empty!" << endl;
```



```

55         cout << "empty!" << endl;
56         return;
57     }
58     cout << "Linked List: ";
59     while (temp != NULL) {
60         cout << temp->data << " → ";
61         temp = temp->next;
62     }
63     cout << "NULL" << endl;
64 }
65 };
66
67 int main() {
68     SinglyLL list;
69     list.insert(10);
70     list.insert(20);
71     list.insert(30);
72     list.display();
73     list.remove(20);
74     list.display();
75     return 0;
}

```

Doubly Linked Lists

```

3 struct Node {
4     int data;
5     Node* next;
6     Node* prev;
7 };
8
9 class doublyLL {
10     Node* head;
11
12 public:
13     doublyLL() { head = NULL; }
14     void insert(int value) {
15         Node* newNode = new Node();
16         newNode->data = value;
17         newNode->next = NULL;
18         newNode->prev = NULL;
19
20         if (head == NULL) {
21             head = newNode;
22         } else {
23             Node* temp = head;
24             while (temp->next != NULL)
25                 temp = temp->next;
26             temp->next = newNode;
27             newNode->prev = temp;
28         }
29         cout << "Inserted: " << value << endl;
30
31     void remove(int value) {
32         if (head == NULL) {
33             cout << "List is empty!" << endl;
34             return;
35         }
36         Node* temp = head;
37
38         while (temp != NULL && temp->data != value)
39             temp = temp->next;
40
41         if (temp == NULL) {
42             cout << "Value not found!" << endl;
43             return;
44         }
45         if (temp->prev != NULL)
46             temp->prev->next = temp->next;
47         else
48             head = temp->next;
49
50         if (temp->next != NULL)
51             temp->next->prev = temp->prev;
52
53         delete temp;
54         cout << "Deleted: " << value << endl;
55     }
}

```

```

56 void display() {
57     Node* temp = head;
58     if (!temp) {
59         cout << "List is empty!" << endl;
60         return;
61     }
62     cout << "Doubly Linked List: ";
63     while (temp != NULL) {
64         cout << temp->data << " ↔ ";
65         temp = temp->next;
66     }
67     cout << "NULL" << endl;
68 }
69 };
70 int main() {
71     doublyLL list;
72     list.insert(10);
73     list.insert(20);
74     list.insert(30);
75     list.display();
76     list.remove(20);
77     list.display();
78     return 0;
79 }
80 |

```

Output:

```

Microsoft Windows [Version 10.0.26100.3476]
(c) Microsoft Corporation. All rights reserved.

D:\sem 2 dsa>cd "d:\sem 2 dsa\" && g++ LinkedLists.cpp -o LinkedLists && "d:\sem 2 dsa\"LinkedLists
Inserted: 10
Inserted: 20
Inserted: 30
Linked List: 10 -> 20 -> 30 -> NULL
Deleted: 20
Linked List: 10 -> 30 -> NULL

d:\sem 2 dsa>

```

```

Microsoft Windows [Version 10.0.26100.3476]
(c) Microsoft Corporation. All rights reserved.

D:\sem 2 dsa>cd "d:\sem 2 dsa\" && g++ doublyLL.cpp -o doublyLL && "d:\sem 2 dsa\"doublyLL
Inserted: 10
Inserted: 20
Inserted: 30
Doubly Linked List: 10 <-> 20 <-> 30 <-> NULL
Deleted: 20
Doubly Linked List: 10 <-> 30 <-> NULL

```

Q5. Write a program to implement a double-ended queue using a linked list

Code:

```
4 struct Node {
5     int data;
6     Node* next;
7     Node* prev;
8 };
9
10 class Deque {
11     Node* front;
12     Node* rear;
13
14 public:
15     Deque() {
16         front = rear = NULL;
17     }
18     void insertFront(int value) {
19         Node* newNode = new Node();
20         newNode->data = value;
21         newNode->next = front;
22         newNode->prev = NULL;
23
24         if (front == NULL) {
25             front = rear = newNode;
26         } else {
27             front->prev = newNode;
28             front = newNode;
29         }
30     }
31     void insertRear(int value) {
32         Node* newNode = new Node();
33         newNode->data = value;
34         newNode->next = NULL;
35         newNode->prev = rear;
36
37         if (rear == NULL) {
38             front = rear = newNode;
39         } else {
40             rear->next = newNode;
41             rear = newNode;
42         }
43         cout << "Inserted at Rear: " << value << endl;
44     }
45
46     // Delete from front
47     void deleteFront() {
48         if (front == NULL) {
49             cout << "Deque Underflow! Cannot delete from front." << endl;
50             return;
51         }
52         Node* temp = front;
53         front = front->next;
54
55         if (front == NULL) rear = NULL; // If deque is empty
56         else front->prev = NULL;
57     }
58
59     void display() {
60         if (front == NULL) {
61             cout << "Deque is empty!" << endl;
62             return;
63         }
64         Node* temp = front;
65         cout << "Deque elements: ";
66         while (temp != NULL) {
67             cout << temp->data << " ↔ ";
68             temp = temp->next;
69         }
70         cout << "NULL" << endl;
71     }
72 };
73
74 int main() {
75     Deque dq;
76     dq.insertFront(10);
77     dq.insertRear(20);
78     dq.insertFront(5);
79     dq.insertRear(30);
80     dq.display();
81
82     dq.deleteFront();
83     dq.deleteRear();
84     dq.display();
85
86     return 0;
87 }
```

Output:

```
Microsoft Windows [Version 10.0.26100.3476]
(c) Microsoft Corporation. All rights reserved.

D:\sem 2 dsa>cd "d:\sem 2 dsa\" && g++ doublyEndedQueue.cpp -o doublyEndedQueue && "d:\sem 2 dsa\"doublyEndedQueue
Inserted at Front: 10
Inserted at Rear: 20
Inserted at Front: 5
Inserted at Rear: 30
Deque elements: 5 <-> 10 <-> 20 <-> 30 <-> NULL
Deleted from Front: 5
Deleted from Rear: 30
Deque elements: 10 <-> 20 <-> NULL
```

Q6. Write a program to construct a binary tree and display its preorder, inorder and postorder traversals.

Code:

```
3 struct Node {
4     int data;
5     Node* left;
6     Node* right;
7
8     Node(int value) {
9         data = value;
10        left = right = NULL;
11    }
12};
13class BinaryTree {
14public:
15    Node* root;
16
17    BinaryTree() { root = NULL; }
18    Node* insert(Node* node, int value) {
19        if (node == NULL) {
20            return new Node(value);
21        }
22        if (value < node->data) {
23            node->left = insert(node->left, value);
24        } else {
25            node->right = insert(node->right, value);
26        }
27        return node;
28    }
```

```
29 void preorder(Node* node) {
30     if (node == NULL) return;
31     cout << node->data << " ";
32     preorder(node->left);
33     preorder(node->right);
34 }
35 void inorder(Node* node) {
36     if (node == NULL) return;
37     inorder(node->left);
38     cout << node->data << " ";
39     inorder(node->right);
40 }
41 void postorder(Node* node) {
42     if (node == NULL) return;
43     postorder(node->left);
44     postorder(node->right);
45     cout << node->data << " ";
46 }
47 };
48 int main() {
49     BinaryTree tree;
50     tree.root = tree.insert(tree.root, 50);
51     tree.insert(tree.root, 30);
52     tree.insert(tree.root, 70);
53     tree.insert(tree.root, 20);
54     tree.insert(tree.root, 40);
55     tree.insert(tree.root, 60);
```

```
58     cout << "Preorder Traversal: ";
59     tree.preorder(tree.root);
60     cout << endl;
61
62     cout << "Inorder Traversal: ";
63     tree.inorder(tree.root);
64     cout << endl;
65
66     cout << "Postorder Traversal: ";
67     tree.postorder(tree.root);
68     cout << endl;
```

Output:

```
Microsoft Windows [Version 10.0.26100.3476]
(c) Microsoft Corporation. All rights reserved.

D:\sem 2 dsa>cd "d:\sem 2 dsa\" && g++ BT.cpp -o BT && "d:\sem 2 dsa\BT
Preorder Traversal: 50 30 20 40 70 60 80
Inorder Traversal: 20 30 40 50 60 70 80
Postorder Traversal: 20 40 30 60 80 70 50

d:\sem 2 dsa>
```