

StackDs.java

```
1  /* Stack java */
2  class Stack {
3      private int maxSize;
4      private int[] stackArray;
5      private int top;
6      public Stack(int size) {
7          this.maxSize = size;
8          this.stackArray = new int[maxSize];
9          this.top = -1;
10     }
11     public void push(int value) {
12         if (top < maxSize - 1) {
13             stackArray[++top] = value;
14             System.out.println("Pushed: " + value);
15         } else {
16             System.out.println("Stack is full. Cannot push " + value);
17         }
18     }
19     public int pop() {
20         if (top >= 0) {
21             int value = stackArray[top--];
22             System.out.println("Popped: " + value);
23             return value;
24         } else {
25             System.out.println("Stack is empty. Cannot pop.");
26             return -1;
27         }
28     }
29
30     public int peek() {
31         if (top >= 0) {
32             return stackArray[top];
33         } else {
34             System.out.println("Stack is empty. Cannot peek.");
35             return -1;
36         }
37     }
38 }
39 public class StackDriver {
40     public static void main(String[] args) {
41         Stack stack = new Stack(5);
42         stack.push(10);
43         stack.push(20);
44         stack.push(30);
45         System.out.println("Top element: " + stack.peek());
46         stack.pop();
47         System.out.println("Top element after pop: " + stack.peek());
48         stack.pop();
49         stack.pop();
```

```
50 |         stack.pop();
51 |     }
52 | }
```

Stack.cpp

```
1  /* Stack In CPP */
2  #include <iostream>
3  using namespace std;
4  class Stack
5  {
6  private:
7      int maxSize;
8      int *stackArray;
9      int top;
10
11 public:
12     Stack(int size)
13     {
14         maxSize = size;
15         stackArray = new int[maxSize];
16         top = -1;
17     }
18     void push(int value)
19     {
20         if (top < maxSize - 1)
21         {
22             stackArray[++top] = value;
23             cout << "Pushed: " << value << endl;
24         }
25         else
26         {
27             cout << "Stack is full. Cannot push " << value << endl;
28         }
29     }
30     int pop()
31     {
32         if (top >= 0)
33         {
34             int value = stackArray[top--];
35             cout << "Popped: " << value << endl;
36             return value;
37         }
38         else
39         {
40             cout << "Stack is empty. Cannot pop." << endl;
41             return -1;
42         }
43     }
44     int peek()
45     {
46         if (top >= 0)
47         {
48             return stackArray[top];
49         }
```

```
50         else
51         {
52             cout << "Stack is empty. Cannot peek." << endl;
53             return -1;
54         }
55     }
56     ~Stack()
57     {
58         delete[] stackArray;
59     }
60 };
61 int main()
62 {
63     Stack stack(5);
64     stack.push(10);
65     stack.push(20);
66     stack.push(30);
67     cout << "Top element: " << stack.peek() << endl;
68     stack.pop();
69     cout << "Top element after pop: " << stack.peek() << endl;
70     stack.pop();
71     stack.pop();
72     stack.pop();
73     return 0;
74 }
```

LinkedList.java

```
1  /* Linked List in Java */
2
3  class SingleLinkedList {
4      public static void main(String args[]) {
5          Node head = new Node(6);
6          head.next = new Node(9);
7          head.next.next = new Node(7);
8          head.printLinkedList(head);
9      }
10 }
11 class Node {
12     int data;
13     Node next;
14     Node(int data) {
15         this.data = data;
16         this.next = null;
17     }
18     void printLinkedList(Node head) {
19         while (head != null) {
20             System.out.println(head.data);
21             head = head.next;
22         }
23     }
24 }
```

LinkedList.cpp

```
1  /* LINKED LIST IN CPP */
2  #include <iostream>
3  using namespace std;
4  class Node
5  {
6  public:
7      int data;
8      Node *next;
9      Node(int data)
10     {
11         this->data = data;
12         this->next = nullptr;
13     }
14     void printLinkedList(Node *head)
15     {
16         while (head != nullptr)
17         {
18             cout << head->data << endl;
19             head = head->next;
20         }
21     }
22 };
23 int main()
24 {
25     Node *head = new Node(6);
26     head->next = new Node(9);
27     head->next->next = new Node(7);
28     head->printLinkedList(head);
29     return 0;
30 }
```

CircularLinkedList.java

```
1  /* Circular Linked List in Java */
2  class CircularLinkedList {
3      public static void main(String args[]) {
4          Node head = new Node(6);
5          Node second = new Node(9);
6          Node third = new Node(7);
7          head.next = second;
8          second.next = third;
9          third.next = head;
10         head.printLinkedList(head);
11     }
12 }
13 class Node {
14     int data;
15     Node next;
16     Node(int data) {
17         this.data = data;
18         this.next = null;
19     }
20     void printLinkedList(Node head) {
21         if (head == null) return;
22         Node current = head;
23         do {
24             System.out.println(current.data);
25             current = current.next;
26         } while (current != head);
27     }
28 }
```

CircularLinkedList.cpp

```
1  /* Circular Linked List CPP */
2  #include <iostream>
3  using namespace std;
4  class Node
5  {
6  public:
7      int data;
8      Node *next;
9      Node(int data)
10     {
11         this->data = data;
12         this->next = nullptr;
13     }
14     void printLinkedList(Node *head)
15     {
16         if (head == nullptr)
17             return;
18         Node *current = head;
19         do
20         {
21             cout << current->data << endl;
22             current = current->next;
23         } while (current != head);
24     }
25 };
26 int main()
27 {
28     Node *head = new Node(6);
29     Node *second = new Node(9);
30     Node *third = new Node(7);
31     head->next = second;
32     second->next = third;
33     third->next = head;
34     head->printLinkedList(head);
35     return 0;
36 }
```