

```
1 //1. Write a recursive and non recursive program to calculate the factorial of a number.
2 #include<iostream>
3 using namespace std;
4
5 long int multiplyNumbers(int n);
6
7 int main()
8 {
9     int n;
10    cout<<"Enter a positive integer";
11    cin>>n;
12    cout<<"Factorial of the number is"<<multiplyNumbers(n)<<endl;
13
14    return 0;
15 }
16
17 long int multiplyNumbers(int n)
18 {
19     if(n>=1)
20         return n*multiplyNumbers(n-1);
21
22     else
23         return 1;
24 }
25
```

```
1  #include<iostream>
2  using namespace std;
3
4  int main()
5  {
6
7      int count;
8      int sum;
9
10     int arr[10]{1,2,3,4,5,6,7,8,9,10};
11
12     for(count=0; count<=9; count++)
13     {
14         sum= sum + arr[count];
15
16     }
17
18     cout<<"Sum of Array elements is= "<<sum;
19
20     return 0;
21 }
```

```
1 //Write a program to perform linear search for a given value in unsorted list.
2 #include<iostream>
3 using namespace std;
4
5 int search(int arr[],int n,int x)
6 {
7     for(int i=0; i<n; i++)
8         if (arr[i]==x)
9             return i;
10
11     return -1;
12 }
13 int main()
14 {
15     int arr[]={2,4,0,1,9};
16     int x=1;
17     int n= sizeof(arr)/ sizeof(arr[0]);
18
19     int result = search(arr, n, x);
20
21     (result == -1) ? cout<<"Element not found"<<endl : cout<<"Element found at index ="<<result<<endl;
22
23     return 0;
24 }
```

```
1  #include <iostream>
2  using namespace std;
3
4  int binarySearch(int array[], int x, int low, int high)
5  {
6      if (high >= low)
7      {
8          int mid = low + (high-low) /2;
9
10
11         if (array[mid] == x)
12             return mid;
13
14
15         if (array[mid]> x)
16             return binarySearch(array, x, low, mid -1);
17
18
19
20         return binarySearch (array, x, mid + 1, high );
21     }
22     return -1;
23 }
24
25 int main(void)
26 {
27     int array[] = {3, 4, 5, 6, 7, 8, 9};
28     int n= sizeof(array)/ sizeof (array [0]);
29     int x = 4;
30     int result = binarySearch(array, x, 0, n - 1);
31
32     if (result == -1)
33         cout<<"Not found";
34
35     else
36
37         cout<<"Element is found at index :"<<result<<endl;
38 }
```

```

1 // 7). Write a program that implements Selection sort, to sort a given list of integers in ascending order.
2
3 #include <iostream>
4 using namespace std;
5
6 // function to swap the the position of two elements
7 void swap(int *a, int *b) {
8     int temp = *a;
9     *a = *b;
10    *b = temp;
11 }
12
13 // function to print an array
14 void printArray(int array[], int size) {
15     for (int i = 0; i < size; i++) {
16         cout << array[i] << " ";
17     }
18     cout << endl;
19 }
20
21 void selectionSort(int array[], int size) {
22     for (int step = 0; step < size - 1; step++) {
23         int min_idx = step;
24         for (int i = step + 1; i < size; i++) {
25
26             // To sort in descending order, change > to < in this line.
27             // Select the minimum element in each loop.
28             if (array[i] < array[min_idx])
29                 min_idx = i;
30         }
31
32         // put min at the correct position
33         swap(&array[min_idx], &array[step]);
34     }
35 }
36
37 // driver code
38 int main() {
39     int data[] = {20, 12, 10, 15, 2};
40     int size = sizeof(data) / sizeof(data[0]);
41     selectionSort(data, size);
42     cout << "Sorted array in Acsending Order:\n";
43     printArray(data, size);
44 }

```

```

1 // 8) Write a program that implement insertion sort a given list of inters in ascending order.
2
3 #include <iostream>
4 using namespace std;
5
6 // Function to print an array
7 void printArray(int array[], int size) {
8     for (int i = 0; i < size; i++) {
9         cout << array[i] << " ";
10    }
11    cout << endl;
12 }
13
14 void insertionSort(int array[], int size) {
15     for (int step = 1; step < size; step++) {
16         int key = array[step];
17         int j = step - 1;
18
19         // Compare key with each element on the left of it until an element smaller than
20         // it is found.
21         // For descending order, change key<array[j] to key>array[j].
22         while (key < array[j] && j >= 0) {
23             array[j + 1] = array[j];
24             --j;
25         }
26         array[j + 1] = key;
27     }
28 }
29
30 // Driver code
31 int main() {
32     int data[] = {9, 5, 1, 4, 3};
33     int size = sizeof(data) / sizeof(data[0]);
34     insertionSort(data, size);
35     cout << "Sorted array in ascending order:\n";
36     printArray(data, size);
37 }

```

```

1  #include <iostream>
2  using namespace std;
3
4  // A linked list node
5  struct Node
6  {
7      int data;
8      struct Node *next;
9  };
10 //insert a new node in front of the list
11 void push(struct Node** head, int node_data)
12 {
13     /* 1. create and allocate node */
14     struct Node* newNode = new Node;
15
16     /* 2. assign data to node */
17     newNode->data = node_data;
18
19     /* 3. set next of new node as head */
20     newNode->next = (*head);
21
22     /* 4. move the head to point to the new node */
23     (*head) = newNode;
24 }
25
26 //insert new node after a given node
27 void insertAfter(struct Node* prev_node, int node_data)
28 {
29     /*1. check if the given prev_node is NULL */
30     if (prev_node == NULL)
31     {
32         cout<<"the given previous node is required,cannot be NULL"; return; }
33
34     /* 2. create and allocate new node */
35     struct Node* newNode =new Node;
36
37     /* 3. assign data to the node */
38     newNode->data = node_data;
39
40     /* 4. Make next of new node as next of prev_node */
41     newNode->next = prev_node->next;
42
43     /* 5. move the next of prev_node as new_node */
44     prev_node->next = newNode;
45 }
46
47 /* insert new node at the end of the linked list */
48 void append(struct Node** head, int node_data)
49 {
50     /* 1. create and allocate node */
51     struct Node* newNode = new Node;
52
53     struct Node *last = *head; /* used in step 5*/
54
55     /* 2. assign data to the node */
56     newNode->data = node_data;
57
58     /* 3. set next pointer of new node to null as its the last node*/
59     newNode->next = NULL;
60
61     /* 4. if list is empty, new node becomes first node */
62     if (*head == NULL)
63     {
64         *head = newNode;
65         return;
66     }

```

```

67
68 /* 5. Else traverse till the last node */
69 while (last->next != NULL)
70     last = last->next;
71
72 /* 6. Change the next of last node */
73 last->next = newNode;
74 return;
75 }
76
77 // display linked list contents
78 void displayList(struct Node *node)
79 {
80     //traverse the list to display each node
81     while (node != NULL)
82     {
83         cout<<node->data<<"-->";
84         node = node->next;
85     }
86
87 if(node== NULL)
88     cout<<"null";
89 }
90 /* main program for linked list*/
91 int main()
92 {
93     /* empty list */
94     struct Node* head = NULL;
95
96     // Insert 10.
97     append(&head, 10);
98
99     // Insert 20 at the beginning.
100    push(&head, 20);
101
102    // Insert 30 at the beginning.
103    push(&head, 30);
104
105    // Insert 40 at the end.
106    append(&head, 40); //
107
108
109    insertAfter(head->next, 50);
110
111    cout<<"Final linked list: "<<endl;
112    displayList(head);
113
114    return 0;
115 }

```



```

1 //10) Write a program that uses functions to perform deletion operation on singly linked list.
2 #include <bits/stdc++.h>
3 using namespace std;
4
5 /* Link list node */
6 class Node {
7 public:
8     int data;
9     Node* next;
10 };
11
12 /* Function to delete the entire linked list */
13 void deleteList(Node** head_ref)
14 {
15
16     /* deref head_ref to get the real head */
17     Node* current = *head_ref;
18     Node* next = NULL;
19
20     while (current != NULL)
21     {
22         next = current->next;
23         free(current);
24         current = next;
25     }
26
27     /* deref head_ref to affect the real head back
28        in the caller. */
29     *head_ref = NULL;
30 }
31
32 /* Given a reference (pointer to pointer) to the head
33 of a list and an int, push a new node on the front
34 of the list. */
35 void push(Node** head_ref, int new_data)
36 {
37     /* allocate node */
38     Node* new_node = new Node();
39
40     /* put in the data */
41     new_node->data = new_data;
42
43     /* link the old list off the new node */
44     new_node->next = (*head_ref);
45
46     /* move the head to point to the new node */
47     (*head_ref) = new_node;
48 }
49
50 /* Driver code*/
51 int main()
52 {
53     /* Start with the empty list */
54     Node* head = NULL;
55
56     /* Use push() to construct below list
57 1->12->1->4->1 */
58     push(&head, 1);
59     push(&head, 4);
60     push(&head, 1);
61     push(&head, 12);
62     push(&head, 1);
63
64     cout << "Deleting linked list";
65     deleteList(&head);
66

```

```
67     cout << "\nLinked list deleted";
68 }
69
70 // This is code is contributed by rathbhupendra
```

```

1 // 11) Write a program that implement stack and its operations(push and pop) using array.
2 #include <iostream>
3 using namespace std;
4
5 int stack[100], n=100, top=-1;
6
7 void push (int val)
8 {
9     if ( top>=n-1 )
10     {
11         cout<<"Stack Overflow "<<endl;
12     }
13
14     else
15     {
16         top++;
17         stack[top]=val;
18     }
19 }
20
21 void pop()
22 {
23     if (top<=-1)
24     {
25         cout<<"Stack Underflow"<<endl;
26     }
27
28     else
29     {
30         cout<<"The popped element is "<< stack[top] <<endl;
31         top--;
32     }
33 }
34
35 void display()
36 {
37     if (top>=0)
38     {
39         cout<<"Stack elements are :";
40
41         for(int i=top; i>=0; i--)
42             cout<<stack[i]<<" ";
43         cout<<endl;
44     }
45
46     else
47     {
48         cout<<"Stack is empty";
49     }
50 }
51
52 int main()
53 {
54     int ch, val;
55     cout<<"1) Push in stack"<<endl;
56     cout<<"2) Pop from stack"<<endl;
57     cout<<"3) Display stack"<<endl;
58     cout<<"4) Exit"<<endl;
59
60     do{
61         cout<<"Enter choice : "<<endl;
62         cin>>ch;
63         switch(ch)
64         {
65             case 1:
66                 {

```

```
67         cout<<"Enter value to be pushed : "<<endl;
68         cin>>val;
69         push(val);
70         break;
71     }
72     case 2:
73     {
74         pop( );
75         break;
76     }
77     case 3:
78     {
79         display();
80         break;
81     }
82     case 4:
83     {
84         cout<<"Exit"<<endl;
85         break;
86     }
87     default:
88     {
89         cout<<"Invalid choice"<<endl;
90     }
91 }
92 }
93 }
94 while(ch!=4);
95 return 0;
96
97 }
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
```

```

1 //12) Write a program that implements stack and its operation( push and pop) using linked list.
2 #include <iostream>
3 using namespace std;
4 struct Node {
5     int data;
6     struct Node *next;
7 };
8 struct Node* top = NULL;
9 void push(int val) {
10     struct Node* newnode = (struct Node*) malloc(sizeof(struct Node));
11     newnode->data = val;
12     newnode->next = top;
13     top = newnode;
14 }
15 void pop() {
16     if(top==NULL)
17         cout<<"Stack Underflow"<<endl;
18     else {
19         cout<<"The popped element is "<< top->data <<endl;
20         top = top->next;
21     }
22 }
23 void display() {
24     struct Node* ptr;
25     if(top==NULL)
26         cout<<"stack is empty";
27     else {
28         ptr = top;
29         cout<<"Stack elements are: ";
30         while (ptr != NULL) {
31             cout<< ptr->data <<" ";
32             ptr = ptr->next;
33         }
34     }
35     cout<<endl;
36 }
37 int main() {
38     int ch, val;
39     cout<<"1) Push in stack"<<endl;
40     cout<<"2) Pop from stack"<<endl;
41     cout<<"3) Display stack"<<endl;
42     cout<<"4) Exit"<<endl;
43     do {
44         cout<<"Enter choice: "<<endl;
45         cin>>ch;
46         switch(ch) {
47             case 1: {
48                 cout<<"Enter value to be pushed:"<<endl;
49                 cin>>val;
50                 push(val);
51                 break;
52             }
53             case 2: {
54                 pop();
55                 break;
56             }
57             case 3: {
58                 display();
59                 break;
60             }
61             case 4: {
62                 cout<<"Exit"<<endl;
63                 break;
64             }
65             default: {
66                 cout<<"Invalid Choice"<<endl;

```

```
67         }
68     }
69     }while(ch!=4);
70     return 0;
71 }
```