



Amity University Uttar Pradesh
(Data Structures using C[CSIT124])

Lab File

By Praveen Pandey
Amity School Of Engineering And Technology
A023119822016
B.Tech. Artificial Intelligence
2022-2026

Submitted to:
Dr Suchi Mala
Amity School of Engineering
And Technology

Index

1	Write a C program that implements a circular queue using an array.
2	Write a C program that allows the user to delete an element from an array.
3	Write a C program that allows the user to insert an element at a specified position in an array.
4	Write a program to check if an expression containing parentheses, curly braces, and square brackets is balanced using a stack.
5	Write a program to implement a queue using an array.
6	Write a program to convert a sparse matrix into its compact matrix representation.
7	Write a program to implement a stack using an array.
8	Write a program to sort an array of integers using the bubble sort algorithm.
9	Write a program to sort an array of integers using the insertion sort algorithm.
10	Write a program to sort an array of integers using the merge sort algorithm.

11	Write a program to sort an array of integers using the quick sort algorithm.
12	Write a program to sort an array of integers using the selection sort algorithm.
13	Write a program to manage a doubly linked list
14	Write a program to manage a queue using a linked list.
15	Write a program to manage a singly linked list
16	Write a C program that implements a stack using a linked list with operations for insertion, deletion, and traversal.
17	Write a C program that performs binary search on a sorted array and calculates the execution time for the search.
18	Write a C program that performs linear search on an array and calculates the execution time for the search.
19	Write a C program that implements a binary search tree (BST) with functions for insertion, deletion, and traversal (in order, preorder, post order).

1. Write a C program that implements a circular queue using an array.

```
1  #include <stdio.h>
2  #define n 5
3  int arr[n];
4  int front=-1, rear=-1;
5
6  void enqueue(int num) {
7      if (front == -1 && rear == -1) {
8          front++;
9          rear++;
10         arr[rear] = num;
11     } else if ((rear + 1) % n == front) {
12         printf("Queue overflow\n");
13     } else {
14         rear = (rear + 1) % n;
15         arr[rear] = num;
16     }
17 }
18
19 void dequeue() {
20     if (front == -1 && rear == -1) {
21         printf("Queue underflow\n");
22     } else if (front == rear) {
23         front = -1;
24         rear = -1;
25     } else {
26         front = (front + 1) % n;
27     }
28 }
29
```

```
30 void display() {
31     if (front == -1 && rear == -1) {
32         printf("Queue is empty\n");
33     } else {
34         int i = front;
35         while (i != rear) {
36             printf("%d\t", arr[i]);
37             i = (i + 1) % n;
38         }
39         printf("%d\t", arr[i]);
40     }
41     printf("\n");
42 }
43
44 void peek() {
45     if (front == -1 && rear == -1) {
46         printf("Queue is empty\n");
47     } else {
48         printf("%d\n", arr[front]);
49     }
50 }
51
52 int main(void) {
53     printf("Welcome\n");
54     char choice = 'y';
55     int select, ele;
56     printf("Enter 1 for enqueue\n");
57     printf("Enter 2 for dequeue\n");
58     printf("Enter 3 for Display\n");
59     printf("Enter 4 for peek\n");
60     while (choice == 'y') {
```

```

61     printf("\nEnter the choice: ");
62     scanf("%d", &select);
63     switch (select) {
64         case 1:
65             printf("Enter the element: ");
66             scanf("%d", &ele);
67             enqueue(ele);
68             break;
69         case 2:
70             dequeue();
71             break;
72         case 3:
73             display();
74             break;
75         case 4:
76             peek();
77             break;
78         default:
79             printf("Wrong choice entered!!\n");
80     }
81     printf("Want to continue(y/n): ");
82     scanf(" %c", &choice);
83 }
84 return 0;
85 }

```

Output

```
/tmp/j0hytmJ62A.o
```

```
Welcome
```

```
Enter 1 for enqueue
```

```
Enter 2 for dequeue
```

```
Enter 3 for Display
```

```
Enter 4 for peek
```

```
Enter the choice: 2
```

```
Queue underflow
```

```
Want to continue(y/n): n
```

```
=== Code Execution Successful ===
```

2. Write a C program that allows the user to delete an element from an array.

```
1  #include <stdio.h>
2  int main() {
3      int n;
4      printf("Enter size of array: ");
5      scanf("%d",&n);
6      int a[n],i,index=-1;
7      printf("Enter the elements: ");
8      for(i=0;i<n;i++) scanf("%d",&a[i]);
9      int item;
10     printf("Enter item to be deleted: ");
11     scanf("%d",&item);
12     //Searching for index of item
13     for(i=0;i<n;i++){
14         if(a[i]==item) index=i;
15     }
16     if (index== -1) printf("Element not present in the array\n");
17     else{
18         n--;
19         for(i=index;i<n;i++) a[i]=a[i+1];
20     }
21     printf("After deletion : ");
22     for(i=0;i<n;i++) printf("%d\t",a[i]);
23     return 0;
24 }
```

Output

```
/tmp/7TOV9SBL8W.o
Enter size of array: 5
Enter the elements: 10 20 30 40 50
Enter item to be deleted: 30
After deletion : 10 20 40 50

=== Code Execution Successful ===
```

3. Write a C program that allows the user to insert an element at a specified position in an array.

```
1  #include <stdio.h>
2  #define N 100
3  int main(void) {
4      int size,ele,pos;
5      printf("Enter the size of array: ");
6      scanf("%d", &size);
7      int arr[N];
8      printf("Enter the elements: ");
9      for(int i=0;i<size;i++) scanf("%d",&arr[i]);
10     printf("Enter the element to insert: ");
11     scanf("%d", &ele);
12     printf("Enter the position: ");
13     scanf("%d", &pos);
14     size++;
15     for(int i=size-1;i>=pos;i--) arr[i]=arr[i-1]; //Shifting the elemtns to the
        right
16     arr[pos-1]=ele;
17     printf("Array after insertion: ");
18     for(int i=0 ;i<size;i++) printf("%d ",arr[i]);
19     return 0;
20 }
```

Output

```
/tmp/yB2bk0Swfa.o
Enter the size of array: 5
Enter the elements: 10 20 30 40 50
Enter the element to insert: 25
Enter the position: 3
Array after insertion: 10 20 25 30 40 50

=== Code Execution Successful ===
```


4. Write a program to check if an expression containing parentheses, curly braces, and square brackets is balanced using a stack.

```
1  #include <stdio.h>
2  #include <stdbool.h>
3  #include <string.h>
4  #define N 100
5  char stack[N];
6  int top=-1;
7  bool isEmpty(){
8      if(top==-1) return true;
9      return false;
10 }
11 bool isFull(){
12     if(top==N-1) return true;
13     return false;
14 }
15 void push(char ele){
16     if(isFull()){
17         printf("Stack overflow\n");
18         return;
19     }
20     stack[++top]=ele;
21 }
22 void pop(){
23     if(isEmpty()){
24         printf("Stack underflow\n");
25         return;
26     }
27     top--;
28 }
29 void display(){
30     if(isEmpty()){
```

```

31     printf("Stack is empty\n");
32     return;
33 }
34 printf("The elements are: ");
35 for(int i=0;i<=top;i++){
36     printf("%d\t",stack[i]);
37 }
38 printf("\n");
39 }
40 int main(void) {
41     char exp[100];
42     printf("Enter the expression: ");
43     fgets(exp, sizeof(exp), stdin);
44     int i=0;
45     for(i=0;i<strlen(exp);i++){
46         if(exp[i]=='['||exp[i]=='{'||exp[i]=='(') push(exp[i]);
47         else if(exp[i]==']'){
48             if(isEmpty()) printf("Unbalanced\n");
49             else if(stack[top]!='['){
50                 printf("Unbalanced\n");
51                 break;
52             }
53             else pop();
54         }
55         else if(exp[i]==')'){
56             if(isEmpty()) printf("Unbalanced\n");
57             else if(stack[top]!='('){
58                 printf("Unbalanced\n");
59                 break;
60             }
61             else pop();
62         }
63         else if(exp[i]=='}') {
64             if(isEmpty()) printf("Unbalanced\n");
65             else if(stack[top]!='{'){
66                 printf("Unbalanced\n");
67                 break;
68             }
69             else pop();
70         }
71     }
72     if(isEmpty()) printf("Expression is balanced!!");
73     return 0;
74 }

```

Output

```
/tmp/JWsbDeza11.o
Enter the expression: {(a+b)*[c/d]}
Expression is balanced!!

=== Code Execution Successful ===
```

5. Write a program to implement a queue using an array.

```
1  #include <stdio.h>
2  #include <stdbool.h>
3  #define size 7
4  int front=-1,rear=-1;
5  int queue[size];
6  bool isFull(){
7      if(rear==size-1) return true;
8      return false;
9  }
10 bool isEmpty(){
11     if (front==-1&&rear==-1) return true;
12     return false;
13 }
14 void enqueue(){
15     if(isFull()){
16         printf("Queue overflow\n");
17         return;
18     }
19     int ele;
20     printf("Enter the element: ");
21     scanf("%d",&ele);
22     if(isEmpty()){
23         front++;
24         rear++;
25         queue[front]=ele;
26     }
27     else queue[++rear]=ele;
28     return;
29 }
30 void dequeue(){
31     if(isEmpty()){
```

```

32     printf("Queue underflow\n");
33     return;
34 }
35 int ele;
36 ▾ if(front==rear){
37     ele=queue[rear];
38     front=rear=-1;
39 }
40 else ele=queue[front++];
41 printf("Deleted element is %d\n",ele);
42 return;
43 }
44 ▾ void display(){
45 ▾ if(isEmpty()){
46     printf("Queue is empty\n");
47     return;
48 }
49 ▾ else{
50     printf("The elements are: ");
51     for(int i=front;i<=rear;i++) printf("%d\t",queue[i]);
52 }
53 printf("\n");
54 return;
55 }
56 ▾ int main(void) {
57     printf("Enter 1 for Enqueue operation\n");
58     printf("Enter 2 for Dequeue operation\n");
59     printf("Enter 3 for Display operation\n");
60     char c='y';
61     int choice;
62 ▾ while(c=='y'){
63     printf("\nEnter the choice: ");
64     scanf(" %d",&choice);
65 ▾ switch(choice){
66     case 1:
67         enqueue();
68         break;
69     case 2:
70         dequeue();
71         break;
72     case 3:
73         display();
74         break;
75     default:
76         printf("Invalid choice!!\n");
77         break;
78     }
79     printf("Want to continue(y/n): ");
80     scanf(" %c",&c);
81 }
82 printf("\nPrograms ends!!!");
83 return 0;
84 }

```

Output

```
/tmp/cCAUj7yMdl.o
Enter 1 for Enqueue operation
Enter 2 for Dequeue operation
Enter 3 for Display operation

Enter the choice: 1
Enter the element: 5
Want to continue(y/n): n

Programs ends!!!

=== Code Execution Successful ===
```

6. Write a program to convert a sparse matrix into its compact matrix representation.

```
1  #include <stdio.h>
2  int main(void) {
3      int rows,columns,size=0;
4      printf("Enter the no. of rows: ");
5      scanf("%d",&rows);
6      printf("Enter the no of columns: ");
7      scanf("%d",&columns);
8      printf("Enter the elements: \n");
9      int sparsematrix[rows][columns];
10     for(int i=0;i<rows;i++){
11         for(int j=0;j<columns;j++){
12             scanf("%d",&sparsematrix[i][j]);
13             if(sparsematrix[i][j]!=0) size++;
14         }
15     }
```

```

16  int compactmatrix[3][size],k=0;
17  for(int i=0;i<rows;i++){
18      for(int j=0;j<columns;j++){
19          if(sparsematrix[i][j]!=0){
20              compactmatrix[0][k]=i;
21              compactmatrix[1][k]=j;
22              compactmatrix[2][k]=sparsematrix[i][j];
23              k++;
24          }
25      }
26  }
27  printf("Sparse matrix representation: \n");
28  for(int i=0;i<3;i++){
29      for(int j=0;j<size;j++){
30          printf("%d\t",compactmatrix[i][j]); //Each columns represent row, column,
          and value
31      }
32      printf("\n");
33  }
34  return 0;
35  }

```

Output

```

/tmp/jxHu6i5zDD.o
Enter the no. of rows: 3
Enter the no of columns: 4
Enter the elements:
0 0 0 0
1 0 0 2
0 3 0 0 0 0 0

1 0 0 2

0 3 0 0
Sparse matrix representation:
1  1  2
0  3  1
1  2  3

=== Code Execution Successful ===

```

7. Write a program to implement a stack using an array.

```
1  #include <stdio.h>
2  #define n 5
3  int arr[n];
4  int front=-1, rear=-1;
5  void enqueue(int num){
6  if (front==-1&&rear==-1){
7      front++;
8      rear++;
9      arr[rear]=num;
10 }
11 else if((rear+1)%n==front) printf("Queue overflow\n");
12 else{
13     rear=(rear+1)%n;
14     arr[rear]=num;
15 }
16 }
17 void dequeue(){
18     if(front==-1&&rear==-1) printf("Queue underflow\n");
19     else if(front==rear){
20         front=-1;
21         rear=-1;
22     }
23     else front=(front+1)%n;
24 }
25 void display(){
26     if(front==-1 && rear==-1) printf("Queue is empty");
27     else{
28         int i=front;
29         while(i!=rear){
30             printf("%d\t", arr[i]);
31             i=(i+1)%n;
32         }
33         printf("%d\t", arr[i]);
34     }
35     printf("\n");
36 }
37 void peek(){
38     if(front==-1&&rear==-1) printf("Queue is empty");
39     else printf("%d", arr[front]);
40     printf("\n");
41 }
42 int main(void) {
43     printf("Welcome\n");
44     char choice='y';
45     int select, ele;
```

```

46  printf("Enter 1 for enqueue\n");
47  printf("Enter 2 for dequeue\n");
48  printf("Enter 3 for Display\n");
49  printf("Enter 4 for peek\n");
50  while(choice=='y'){
51      printf("\nEnter the choice: ");
52      scanf("%d",&select);
53      switch(select){
54          case 1:
55              printf("Enter the element: ");
56              scanf("%d",&ele);
57              enqueue(ele);
58              break;
59          case 2:
60              dequeue();
61              break;
62          case 3:
63              display();
64              break;
65          case 4:
66              peek();
67              break;
68          default:
69              printf("Wrong choice entered!!\n");
70      }
71      printf("Want to continue(y/n): ");
72      scanf(" %c",&choice);
73  }
74  return 0;
75 }

```

Output

```
/tmp/zL4DZNKoDj.o
```

```
Welcome
```

```
Enter 1 for enqueue
```

```
Enter 2 for dequeue
```

```
Enter 3 for Display
```

```
Enter 4 for peek
```

```
Enter the choice: 1
```

```
Enter the element: 10
```

```
Want to continue(y/n): n
```

```
=== Code Execution Successful ===
```


8. Write a program to sort an array of integers using the bubble sort algorithm.

```
1  #include <stdio.h>
2  int swap(int *a,int *b){
3      int temp=*a;
4      *a=*b;
5      *b=temp;
6  }
7  int main(void) {
8      int n;
9      printf("Enter the size: ");
10     scanf("%d",&n);
11     int arr[n];
12     printf("Enter the array: ");
13     for(int i=0;i<n;i++){
14         scanf("%d",&arr[i]);
15     }
16     int j=n;
17     while(j>=0){
18         for(int i=0;i<j-1;i++){
19             if(arr[i]>arr[i+1]) swap(&arr[i],&arr[i+1]);
20         }
21         j--;
22     }
23     printf("Sorted array is: ");
24     for(int i=0;i<n;i++){
25         printf("%d\t",arr[i]);
26     }
27     return 0;
28 }
```

Output

```
/tmp/kSfuC404SX.o
Enter the size: 5
Enter the array: 64 34 25 12 22
Sorted array is: 12 22 25 34 64

=== Code Execution Successful ===
```

9. Write a program to sort an array of integers using the insertion sort algorithm.

```
1  #include<stdio.h>
2  void swap(int *a,int*b){
3      int temp=*a;
4      *a=*b;
5      *b=temp;
6  }
7  int main(void) {
8      int n;
9      printf("Enter the number of elements: ");
10     scanf("%d",&n);
11     int arr[n];
12     printf("Enter the elements: ");
13     for(int i=0;i<n;i++) scanf("%d",&arr[i]);
14     for(int i=1;i<n;i++){
15         for(int j=i;j>0;j--){
16             if(arr[j-1]<arr[j]) break;
17             else swap(&arr[j],&arr[j-1]);
18             //if(arr[j-1]>arr[j]) swap(&arr[j],&arr[j-1]); I have replaced this line
                with above two lines of code
19         }
20     }
21     printf("Element after sorting: \n");
22     for(int i=0;i<n;i++){
23         printf("%d\t",arr[i]);
24     }
25     return 0;
26 }
```

Output

```
/tmp/goT37114nW.o
Enter the number of elements: 5
Enter the elements: 9 7 5 11 12
Element after sorting:
5   7   9   11  12

=== Code Execution Successful ===
```

10. Write a program to sort an array of integers using the merge sort algorithm.

```
1  #include <stdio.h>
2  void merge(int arr[], int s, int mid, int e) {
3      int n1 = mid - s + 1, n2 = e - mid;
4      int L[n1], R[n2];
5      for(int i = 0; i < n1; i++) L[i] = arr[s + i];
6      for(int i = 0; i < n2; i++) R[i] = arr[mid + i + 1];
7      int i = 0, j = 0, k = s;
8      while(i < n1 && j < n2) {
9          if(L[i] < R[j]) {
10             arr[k] = L[i];
11             i++;
12         }
13         else{
14             arr[k] = R[j];
15             j++;
16         }
17         k++;
18     }
19     while(i < n1) {
20         arr[k] = L[i];
21         i++;
22         k++;
23     }
24     while(j < n2) {
25         arr[k] = R[j];
26         j++;
27         k++;
28     }
29     return;
30 }

31 void mergesort(int arr[], int s, int e) {
32     if (s >= e) return;
33     else{
34         int mid = s + (e - s) / 2;
35         mergesort(arr, s, mid);
36         mergesort(arr, mid + 1, e);
37         merge(arr, s, mid, e);
38     }
39 }

40 int main(void) {
41     int size;
42     printf("Enter the size: ");
43     scanf("%d", &size);
44     int arr[size];
45     printf("Enter the elements: ");
```

```

46     for(int i = 0; i < size; i++) scanf("%d", &arr[i]);
47     mergesort(arr, 0, size - 1);
48     printf("After sorting: ");
49     for(int i = 0; i < size; i++) printf("%d\t", arr[i]);
50     return 0;
51 }

```

Output

```

/tmp/FZS6n4EiTQ.o
Enter the size: 6
Enter the elements: 38 27 43 3 9 82
After sorting: 3    9    27   38   43   82

=== Code Execution Successful ===

```

11. Write a program to sort an array of integers using the quick sort algorithm.

```

1  #include <stdio.h>
2  void swap(int *f,int*s){
3      int temp=*f;
4      *f=*s;
5      *s=temp;
6  }
7  int partion(int arr[],int s,int e){
8      int i=s,j=e;
9      int pivot=s;
10     while(i<j){
11         while(arr[i]<=arr[pivot]&& i<e) i++;
12         while(arr[j]>arr[pivot]&& j>s) j--;
13         if(i<j) swap(&arr[i],&arr[j]);
14     }
15     swap(&arr[pivot],&arr[j]);
16     return j;
17 }
18 void quicksort(int arr[],int s,int e){
19     if(s<e){
20         int p=partion(arr,s,e);
21         quicksort(arr,s,p-1);
22         quicksort(arr,p+1,e);
23     }
24 }
25 int main(){
26     int size;
27     printf("Enter the size of the array: ");
28     scanf("%d",&size);
29     int arr[size];
30     printf("Enter the elements: ");
31     for(int i=0;i<size;i++) scanf("%d",&arr[i]);

```

```

32     quicksort(arr,0,size-1);
33     printf("Elements after sorting: ");
34     for(int i=0;i<size;i++) printf("%d\t",arr[i]);
35     return 0;
36 }

```

Output

```

/tmp/eozgM8cNk7.o
Enter the size of the array: 5
Enter the elements: 10 7 8 9 1
Elements after sorting: 1    7    8    9    10

=== Code Execution Successful ===

```

12. Write a program to sort an array of integers using the selection sort algorithm.

```

1  #include <stdio.h>
2  void swap(int *a,int*b){
3      int temp=*a;
4      *a=*b;
5      *b=temp;
6  }
7  int main(void) {
8      int n;
9      printf("Enter size of array: ");
10     scanf("%d",&n);
11     int arr[n];
12     printf("Enter elements: ");
13     for(int i=0;i<n;i++) scanf("%d",&arr[i]);
14     for(int i=0;i<n;i++){
15         int min=i;
16         for(int j=i;j<n;j++){
17             if(arr[min]>arr[j]) min =j;
18         }
19         swap(&arr[min],&arr[i]);
20     }
21     for(int i=0;i<n;i++) printf("%d\t",arr[i]);
22     return 0;
23 }

```

Output

```
/tmp/I0obORT5zi.o
Enter size of array: 4
Enter elements: 64 25 12 22
12  22  25  64

=== Code Execution Successful ===
```

13. Write a program to manage a doubly linked list

```
1  #include <stdio.h>
2  #include<stdlib.h>
3  struct node{
4      int data;
5      struct node*next,*prev;
6  };
7  void InsertAtHead(struct node**head){
8      int d;
9      printf("Enter the data: ");
10     scanf("%d",&d);
11     struct node* temp;
12     temp=(struct node*)(malloc(sizeof(struct node)));
13     temp->data=d;
14     temp->prev=NULL;
15     if(*head==NULL){
16         temp->next=NULL;
17         *head=temp;
18     }
19     else{
20         temp->next=(*head);
21         *head=temp;
22     }
23     return;
24 }
25 void InsertAtTail(struct node**head){
26     if(*head==NULL){
27         InsertAtHead(head);
28         return;
29     }
30     struct node* newnode;
31     struct node* temp;
```

```

32  newnode=(struct node*)(malloc(sizeof(struct node)));
33  int element;
34  printf("Enter the element: ");
35  scanf("%d",&element);
36  newnode->data=element;
37  newnode->next=NULL;
38  temp=*head;
39  while(temp->next!=NULL){
40      temp=temp->next;
41  }
42  temp->next=newnode;
43  newnode->prev=temp;
44  return;
45  }

```

```

46  void deletion(struct node**head){
47  if(*head==NULL){
48      printf("List is empty\n");
49      return;
50  }
51  int element;
52  printf("Enter the element to delete: ");
53  scanf("%d",&element);
54  struct node*prev=NULL,*curr=*head;
55  if(curr->data==element){//Handling first element
56      *head=(*head)->next;
57      free(curr);
58      return;
59  }
60  while(curr!=NULL&&curr->data!=element){
61      prev=curr;
62      curr=curr->next;
63  }
64  if(curr==NULL){
65      printf("Element not found!!!\n");
66      return;
67  }
68  prev->next=curr->next;
69  curr->next->prev=prev;
70  free(curr);
71  return;
72  }
73  void display(struct node *head){
74  if(head==NULL){
75      printf("Linked list has no elements!!!\n");
76      return;

```

```

77     }
78     printf("The elements are: ");
79     while(head!=NULL){
80         printf("%d\t",head->data);
81         head=head->next;
82     }
83     printf("\n");
84 }
85 void search(struct node* head){
86     int element,pos=1,check=0;
87     printf("Enter the element to search: ");
88     scanf("%d",&element);
89     while(head!=NULL){
90         if(head->data==element){
91             printf("Element found at position %d\n",pos);
92             check=1;
93             return;
94         }
95         head=head->next;
96         pos++;
97     }
98     if(check==0){
99         printf("Element not found!!\n");
100         return;
101     }
102 }
103 int main(void) {
104     printf("Welcome\n");
105     struct node *head=NULL;
106     char ch='y';
107     int choice;

```

14. Write a program to manage a queue using a linked list.

```

1  #include <stdio.h>
2  #include<stdlib.h>
3  struct node{
4      int data;
5      struct node *next;
6  };
7  struct node *front=NULL,*rear=NULL;
8  void enqueue(){
9      struct node* temp=(struct node*)malloc(sizeof(struct node));
10     printf("Enter the data: ");
11     scanf("%d",&temp->data);
12     temp->next=NULL;
13     if(front==NULL&&rear==NULL){
14         rear=front=temp;
15     }
16     else{
17         rear->next=temp;
18         rear=temp;
19     }
20 }

```



```

21 void dequeue(){
22     struct node* temp=front;
23     if(front==NULL &&rear==NULL){
24         printf("Queue underflow\n");
25         return;
26     }
27     else if(front==rear){
28         front=rear=NULL;
29     }
30     else{
31         front=front->next;
32     }
33     free(temp);
34 }
35 void display(){
36     if(front==NULL && rear==NULL) printf("Queue is empty");
37     else{
38         struct node* temp=front;
39         while(temp!=NULL){
40             printf("%d\t",temp->data);
41             temp=temp->next;
42         }
43     }
44     printf("\n");
45 }
46 void peek(){
47     if (front==NULL&& rear==NULL) printf("Queue is empty\n");
48     else printf("%d\n",front->data);
49 }
50 int main(void) {
51     char choice='v':
52     printf("Enter 1 for enqueue\n");
53     printf("Enter 2 for dequeue\n");
54     printf("Enter 3 for display\n");
55     printf("Enter 4 for peek\n");
56     while(choice=='y'){
57         int select;
58         printf("\nEnter the choice: ");
59         scanf("%d",&select);
60         switch(select){
61             case 1:
62                 enqueue();
63                 break;
64             case 2:
65                 dequeue();
66                 break;
67             case 3:
68                 display();
69                 break;
70             case 4:

```

```

71     peek();
72     break;
73     default:
74         printf("Wrong choice entered\n");
75     }
76     printf("Want to continue(y/n): ");
77     scanf(" %c",&choice);
78 }
79 printf("Sucessfully executed\n");
80 return 0;
81 }

```

Output

```

/tmp/lvgMD7p9Kk.o
Enter 1 for enqueue
Enter 2 for dequeue
Enter 3 for display
Enter 4 for peek

Enter the choice: 1
Enter the data: 10
Want to continue(y/n): y

Enter the choice: 1
Enter the data: 20
Want to continue(y/n): y

Enter the choice: 3
10 20
Want to continue(y/n): y

Enter the choice: 4
10
Want to continue(y/n): y

Enter the choice: 2
Want to continue(y/n): y

Enter the choice: 3
20
Want to continue(y/n): n
Sucessfully executed

```

```
108     printf("Enter 1 for insertion at beginning\n");
109     printf("Enter 2 for insertion at tail\n");
110     printf("Enter 3 for deletion\n");
111     printf("Enter 4 for display\n");
112     printf("Enter 5 for searching\n");
113     while(ch=='y'){
114         printf("Enter the choice: ");
115         scanf("%d",&choice);
116         switch(choice){
117             case 1:
118                 InsertAtHead(&head);
119                 break;
120             case 2:
121                 InsertAtTail(&head);
122                 break;
123             case 3:
124                 deletion(&head);
125                 break;
126             case 4:
127                 display(head);
128                 break;
129             case 5:
130                 search(head);
131                 break;
132             default:
133                 printf("Enter choice entered!!\n");
134         }
135         printf("\nWant to continue(y/n): ");
136         scanf(" %c",&ch);
137     }
138     printf("Program executed successfully!!!");
```

```
140 }
```

Output

```
/tmp/4S6I40JdKE.o
Welcome
Enter 1 for insertion at beginning
Enter 2 for insertion at tail
Enter 3 for deletion
Enter 4 for display
Enter 5 for searching
Enter the choice: 1
Enter the data: 10

Want to continue(y/n): y
Enter the choice: 2
Enter the element: 20

Want to continue(y/n): y
Enter the choice: 4
The elements are: 10    20

Want to continue(y/n): y
Enter the choice: 5
Enter the element to search: 10
Element found at position 1
|
Want to continue(y/n): y
Enter the choice: 4
The elements are: 10    20

Want to continue(y/n): n
Program executed successfully!!!

=== Code Execution Successful ===
```

15. Write a program to manage a singly linked list

```
1  #include <stdio.h>
2  #include<stdlib.h>
3  struct node{
4      int data;
5      struct node*next;
6  };
7  void InsertAtHead(struct node**head){
8      int d;
9      printf("Enter the data: ");
10     scanf("%d",&d);
11     struct node* temp;
12     temp=(struct node*)(malloc(sizeof(struct node)));
13     temp->data=d;
14     if(*head==NULL){
15         temp->next=NULL;
16         *head=temp;
17     }
18     else{
19         temp->next=( *head);
20         *head=temp;
21     }
22     return;
23 }
24 void InsertAtTail(struct node**head){
25     if(*head==NULL){
26         InsertAtHead(head);
27         return;
28     }
29     struct node* newnode;
30     struct node* temp;
31     newnode=(struct node*)(malloc(sizeof(struct node)));
32     int element;
33     printf("Enter the element: ");
34     scanf("%d",&element);
35     newnode->data=element;
36     newnode->next=NULL;
37     temp=*head;
38     while(temp->next!=NULL){
39         temp=temp->next;
40     }
41     temp->next=newnode;
42     return;
43 }
44 void InsertAtPos(struct node **head){
45     if(*head==NULL){
46         InsertAtHead(head);
47         return;
48     }
49     struct node *temp=*head;
50     int pos,t=1,element;
```

```

51     printf("Enter the position to enter: ");
52     scanf("%d",&pos);
53     while(temp->next!=NULL&&t<pos-1){
54         temp=temp->next;
55         t++;
56     }
57     printf("Enter the element: ");
58     scanf("%d",&element);
59     struct node* newnode;
60     newnode=(struct node *)(malloc(sizeof(struct node)));
61     newnode->data=element;
62     newnode->next=temp->next;
63     temp->next=newnode;
64 }
65 void deletion(struct node**head){
66     if(*head==NULL){
67         printf("List is empty\n");
68         return;
69     }
70     int element;
71     printf("Enter the element to delete: ");
72     scanf("%d",&element);
73     struct node*prev=NULL,*curr=*head;
74     if(curr->data==element){//Handling first element
75         *head=(*head)->next;
76         free(curr);
77         return;
78     }
79     while(curr!=NULL&&curr->data!=element){
80         prev=curr;
81         curr=curr->next;
82     }
83     if(curr==NULL){
84         printf("Element not found!!!\n");
85         return;
86     }
87     prev->next=curr->next;
88     free(curr);
89     return;
90 }
91 void display(struct node *head){
92     if(head==NULL){
93         printf("Linked list has no elements!!!\n");
94         return;
95     }

```

```

96     printf("The elements are: ");
97     while(head!=NULL){
98         printf("%d\t",head->data);
99         head=head->next;
100     }
101     printf("\n");
102 }
103 void search(struct node* head){
104     int element,pos=1,check=0;
105     printf("Enter the element to search: ");
106     scanf("%d",&element);
107     while(head!=NULL){
108         if(head->data==element){
109             printf("Element found at position %d\n",pos);
110             check=1;
111             return;
112         }
113         head=head->next;
114         pos++;
115     }
116     if(check==0){
117         printf("Element not found!!\n");
118         return;
119     }
120 }
121 int main(void) {
122     printf("Welcome\n");
123     struct node *head=NULL;
124     char ch='y';
125     int choice;
126     printf("Enter 1 for insertion at beginning\n");
127     printf("Enter 2 for insertion at tail\n");
128     printf("Enter 3 for insertion at pos\n");
129     printf("Enter 4 for deletion\n");
130     printf("Enter 5 for display\n");
131     printf("Enter 6 for searching\n");
132     while(ch=='y'){
133         printf("Enter the choice: ");
134         scanf("%d",&choice);
135         switch(choice){
136             case 1:
137                 InsertAtHead(&head);
138                 break;
139             case 2:
140                 InsertAtTail(&head);

```

```

141         break;
142     case 3:
143         InsertAtPos(&head);
144         break;
145     case 4:
146         deletion(&head);
147         break;
148     case 5:
149         display(head);
150         break;
151     case 6:
152         search(head);
153         break;
154     default:
155         printf("Enter choice entered!!\n");
156     }
157     printf("\nWant to continue(y/n): ");
158     scanf(" %c",&ch);
159 }
160 printf("Program executed successfully!!!");
161 return 0;
162 }

```

Output

```

/tmp/yIJQuYRzL.o
Welcome
Enter 1 for insertion at beginning
Enter 2 for insertion at tail
Enter 3 for insertion at pos
Enter 4 for deletion
Enter 5 for display
Enter 6 for searching
Enter the choice: 1
Enter the data: 10

Want to continue(y/n): y
Enter the choice: 1
Enter the data: 20

Want to continue(y/n): y
Enter the choice: 2
Enter the element: 30

Want to continue(y/n): y
Enter the choice: 3
Enter the position to enter: 2
Enter the element: 25

```



```

Want to continue(y/n): y
Enter the choice: 5
The elements are: 20    25  10  30

Want to continue(y/n): y
Enter the choice: 4
Enter the element to delete: 20

Want to continue(y/n): y
Enter the choice: 5
The elements are: 25    10  30

Want to continue(y/n): n
Program executed successfully!!!

=== Code Execution Successful ===

```

16. Write a C program that implements a stack using a linked list with operations for insertion, deletion, and traversal.

```

1  #include <stdio.h>
2  #include<stdlib.h>
3  struct node{
4      int data;
5      struct node* next;
6  };
7  struct node* insert(struct node** top){
8      struct node* newnode;
9      newnode=(struct node*)malloc(sizeof(struct node));
10     int element;
11     printf("Enter the element: ");
12     scanf("%d",&element);
13     newnode->data=element;
14     newnode->next=*top;
15     *top=newnode;
16     return *top;
17 }
18 struct node* deletion(struct node** top){
19     if (*top==NULL){
20         printf("Stack underflow!!!\n");

```

```

21     return *top;
22 }
23 struct node* newnode;
24 newnode=*top;
25 *top=(*top)->next;
26 free(newnode);
27 return *top;
28 }
29 void display(struct node* top){
30     if(top==NULL){
31         printf("Stack is empty!!!\n");
32         return;
33     }
34     printf("Elements are: ");
35     while(top!=NULL){
36         printf("%d\t",top->data);
37         top=top->next;
38     }
39     printf("\n");
40     return;
41 }
42 int main(void) {
43     struct node *top=NULL;
44     char ch='y';
45     int choice;
46     printf("Enter 1 for insertion\n");
47     printf("Enter 2 for deletion\n");
48     printf("Enter 3 for traversal\n");
49     while(ch=='y'){
50         printf("Enter the choice: ");
51         scanf("%d",&choice);
52         switch(choice){
53             case 1:
54                 insert(&top);
55                 break;
56             case 2:
57                 deletion(&top);
58                 break;
59             case 3:
60                 display(top);
61                 break;
62             default:
63                 printf("Invalid choice entered!!!\n");
64         }
65         printf("\nWant to continue(y/n): ");
66         scanf(" %c",&ch);
67     }
68     printf("Program executed succesfully!!!");
69     return 0;
70 }

```

Output

```
/tmp/pERzMiqf3l.o
Enter 1 for insertion
Enter 2 for deletion
Enter 3 for traversal
Enter the choice: 1
Enter the element: 10

Want to continue(y/n): y
Enter the choice: 1
Enter the element: 20

Want to continue(y/n): y
Enter the choice: 3
Elements are: 20    10

Want to continue(y/n): y
Enter the choice: 2

Want to continue(y/n): y
Enter the choice: 2

Want to continue(y/n): n
Program executed succesfully!!!

=== Code Execution Successful ===|
```

17. Write a C program that performs binary search on a sorted array and calculates the execution time for the search.

```
1  #include <stdio.h>
2  #include<time.h>
3  int binarysearch(int ele,int n,int *arr){
4      int s=0,e=n-1;
5      int mid=s+(e-s)/2;
6      while(s<=e){
7          if(arr[mid]==ele) return mid;
8          else if(arr[mid]>ele) e=mid-1;
9          else if(arr[mid]<ele) s=mid+1;
10         mid=s+(e-s)/2;
11     }
12     return -1;
13 }
14 int main(void) {
15     int n;
16     printf("Enter the number of elements: ");
17     scanf("%d",&n);
18     int arr[n];
19     printf("Enter the sorted element: ");
20     for(int i=0;i<n;i++){
21         scanf("%d",&arr[i]);
22     }
23     time_t start,end;
24     start=time(NULL);
25     int ele,index;
26     printf("Enter the element to search: ");
27     scanf("%d",&ele);
28     index=binarysearch(ele,n,arr);
29     if(index==-1) printf("Element not found!!\n");
30     else printf("Element occur at index %d\n",index);
31     end=time(NULL);
32     printf("Execution time for binary search is : %f seconds",difftime(end,start));
33     return 0;
34 }
```

Output

```
/tmp/jI3G7CrRow.o
Enter the number of elements: 5
Enter the sorted element: 1 2 3 4 5
Enter the element to search: 3
Element occur at index 2
Execution time for binary search is : 5.000000 seconds
```

=== Code Execution Successful ===

18. Write a C program that performs linear search on an array and calculates the execution time for the search.

```
1  #include <stdio.h>
2  #include<time.h>
3  int linearsearch(int ele,int n,int *arr){
4      for(int i=0;i<n;i++){
5          if(arr[i]==ele) return i;
6      }
7      return -1;
8  }
9  int main(void) {
10     int n;
11     printf("Enter the number of elements: ");
12     scanf("%d",&n);
13     int arr[n];
14     printf("Enter the element: ");
15     for(int i=0;i<n;i++){
16         scanf("%d",&arr[i]);
17     }
18     time_t start,end;
19     start=time(NULL);
20     int ele;
21     printf("Enter the element to search: ");
22     scanf("%d",&ele);
23     int index=linearsearch(ele,n,arr);
24     if(index==-1) printf("Element not found!!\n");
25     else printf("Element occur at index %d\n",index);
26     end=time(NULL);
27     printf("Execution time for linear search is : %f seconds\n",difftime(end,start));
28     return 0;
29 }
```

Output

```
/tmp/WGUHkRM51S.o
Enter the number of elements: 5
Enter the element: 10 20 30 40 50
Enter the element to search: 30
Element occur at index 2
Execution time for linear search is : 20.000000 seconds
```

=== Code Execution Successful ===

19. Write a C program that implements a binary search tree (BST) with functions for insertion, deletion, and traversal (in order, preorder, post order).

```
1  #include <stdio.h>
2  #include<stdlib.h>
3  struct node{
4      int data;
5      struct node* left,*right;
6  };
7  struct node* createnode(int val){
8      struct node* root=(struct node*)malloc(sizeof(struct node));
9      root->data=val;
10     root->left=NULL;
11     root->right=NULL;
12     return root;
13 }
14 void inorder(struct node *root){
15     if(root!=NULL){
16         inorder(root->left);
17         printf("%d\t",root->data);
18         inorder(root->right);
19     }
20 }
21 void preorder(struct node *root){
22     if(root!=NULL){
23         printf("%d\t",root->data);
24         preorder(root->left);
25         preorder(root->right);
26     }
27 }
28 void postorder(struct node *root){
29     if(root!=NULL){
30         postorder(root->left);
31         postorder(root->right);
32         printf("%d\t",root->data);
33     }
34 }
35 void insert(struct node**root,int ele){
36     if((*root)==NULL){
37         (*root)=createnode(ele);
38         return;
39     }
40     if((*root)->data<ele) insert(&((*root)->right),ele);;
41     if((*root)->data>ele) insert(&((*root)->left),ele);
42 }
43 struct node* nextinorder(struct node**root){
44     struct node*curr=*root;
45     while(curr&&curr->left!=NULL)
```

```

46     curr=curr->left;
47     return curr;
48 }
49 void delete(struct node**root,int ele){
50     if((*root)==NULL) printf("\nElement not found!!\n");
51     else if((*root)->data<ele) delete(&((*root)->right),ele);
52     else if((*root)->data>ele) delete(&((*root)->left),ele);
53     else if((*root)->left&&(*root)->right){
54         struct node* next=nextinorder(&((*root)->right));
55         (*root)->data=next->data;
56         delete(&((*root)->right),next->data);
57     }
58     else{
59         struct node**temp=root;
60         if((*root)->left==NULL&&(*root)->right==NULL) (*root)=NULL;
61         else if((*root)->left!=NULL) (*root)=(*root)->left;
62         else if((*root)->right!=NULL) (*root)=(*root)->right;
63         free(*temp);
64     }
65 }
66 int main(void) {
67     printf("Welcome\n");
68     char ch='y';
69     int ele;
70     struct node *root=NULL;
71     printf("Enter 1 for insertion\n");
72     printf("Enter 2 for deletion\n");
73     printf("Enter 3 for inorder traversal\n");
74     printf("Enter 4 for preorder traversal\n");
75     printf("Enter 5 for postorder traversal\n");
76     while(ch=='y'){
77         int choice;
78         printf("Enter the choice: ");
79         scanf("%d",&choice);
80         switch(choice){
81             case 1:
82                 printf("Enter element to insert: ");
83                 scanf("%d",&ele);
84                 insert(&root,ele);
85                 break;
86             case 2:
87                 printf("Enter element to delete: ");
88                 scanf("%d",&ele);
89                 delete(&root,ele);
90                 break;

```

```

91     case 3:
92         inorder(root);
93         break;
94     case 4:
95         preorder(root);
96         break;
97     case 5:
98         postorder(root);
99         break;
100    default:
101        printf("Invalid choice entered!!!\n");
102    }
103    printf("\nDo you want to continue(y/n): ");
104    scanf(" %c",&ch);
105 }
106 printf("Program Executed successfully!!");
107 return 0;
108 }

```

Output

```

/tmp/AGAAxHdFye.o
Welcome
Enter 1 for insertion
Enter 2 for deletion
Enter 3 for inorder traversal
Enter 4 for preorder traversal
Enter 5 for postorderorder traversal
Enter the choice: 1
Enter element to insert: 10

Do you want to continue(y/n): y
Enter the choice: 1
Enter element to insert: 5

Do you want to continue(y/n): y
Enter the choice: 1
Enter element to insert: 20

Do you want to continue(y/n): y
Enter the choice: 3
5  10  20
Do you want to continue(y/n): y
Enter the choice: 2
Enter element to delete: 5

```


Do you want to continue(y/n): y

Enter the choice: 3

10 20

Do you want to continue(y/n): y

Enter the choice: 4

10 20

Do you want to continue(y/n): y

Enter the choice: 5

20 10

Do you want to continue(y/n): n

Program Executed successfully!!

=== Code Execution Successful ===|