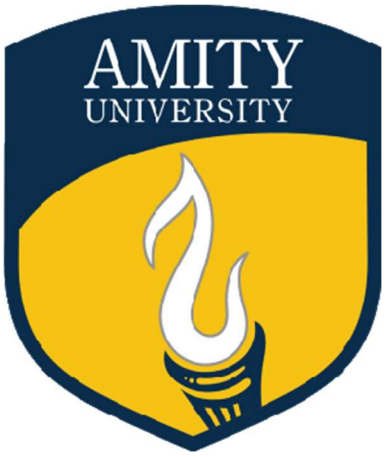

Amity University Uttar Pradesh

(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

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.

```c
#include <stdio.h>
#define n 5
int arr[n];
int front=-1, rear=-1;

void enqueue(int num) {
    if (front == -1 && rear == -1) {
        front++;
        rear++;
        arr[rear] = num;
    } else if ((rear + 1) % n == front) {
        printf("Queue overflow\n");
    } else {
        rear = (rear + 1) % n;
        arr[rear] = num;
    }
}

void dequeue() {
    if (front == -1 && rear == -1) {
        printf("Queue underflow\n");
    } else if (front == rear) {
        front = -1;
        rear = -1;
    } else {
        front = (front + 1) % n;
    }
}

```

```c
void display() {
    if (front == -1 && rear == -1) {
        printf("Queue is empty\n");
    } else {
        int i = front;
        while (i != rear) {
            printf("%d\t", arr[i]);
            i = (i + 1) % n;
        }
        printf("%d\t", arr[i]);
    }
    printf("\n");
}

void peek() {
    if (front == -1 && rear == -1) {
        printf("Queue is empty\n");
    } else {
        printf("%d\n", arr[front]);
    }
}

int main(void) {
    printf("Welcome\n");
    char choice = 'y';
    int select, ele;
    printf("Enter 1 for enqueue\n");
    printf("Enter 2 for dequeue\n");
    printf("Enter 3 for Display\n");
    printf("Enter 4 for peek\n");
    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.

```c
#include <stdio.h>
int main() {
  int n;
  printf("Enter size of array: ");
  scanf("%d",&n);
  int a[n],i,index=-1;
  printf("Enter the elements: ");
  for(i=0;i<n;i++) scanf("%d",&a[i]);
  int item;
  printf("Enter item to be deleted: ");
  scanf("%d",&item);
  //Searching for index of item
  for(i=0;i<n;i++){
    if(a[i]==item) index=i;
  }
  if (index==-1) printf("Element not present in the array\n");
  else{
    n--;
    for(i=index;i<n;i++) a[i]=a[i+1];
  }
  printf("After deletion : ");
  for(i=0;i<n;i++) printf("%d\t",a[i]);
  return 0;
}
```

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.

```c
#include <stdio.h>
#define N 100
int main(void) {
  int size,ele,pos;
  printf("Enter the size of array: ");
  scanf("%d", &size);
  int arr[N];
  printf("Enter the elements: ");
  for(int i=0;i<size;i++) scanf("%d",&arr[i]);
  printf("Enter the element to insert: ");
  scanf("%d", &ele);
  printf("Enter the position: ");
  scanf("%d", &pos);
  size++;
  for(int i=size-1;i>=pos;i--) arr[i]=arr[i-1];//Shifting the elemtns to the
      right
  arr[pos-1]=ele;
  printf("Array after insertion: ");
  for(int i=0 ;i<size;i++) printf("%d ",arr[i]);
  return 0;
}
```

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.

```c
#include <stdio.h>
#include <stdbool.h>
#include<string.h>
#define N 100
char stack[N];
int top=-1;
bool isEmpty(){
  if(top==-1) return true;
  return false;
}
bool isFull(){
  if(top==N-1) return true;
  return false;
}
void push(char ele){
  if(isFull()){
    printf("Stack overflow\n");
    return;
  }
  stack[++top]=ele;
}
void pop(){
  if(isEmpty()){
    printf("Stack underflow\n");
    return;
  }
  top--;
}
void display(){
  if(isEmpty()){
```

```
    printf("Stack is empty\n");
    return;
  }
  printf("The elements are: ");
  for(int i=0;i<=top;i++){
    printf("%d\t",stack[i]);
  }
  printf("\n");
}
int main(void) {
  char exp[100];
  printf("Enter the expression: ");
  fgets(exp, sizeof(exp), stdin);
  int i=0;
  for(i=0;i<strlen(exp);i++){
    if(exp[i]=='['||exp[i]=='{'||exp[i]=='(') push(exp[i]);
    else if(exp[i]==']'){
      if(isEmpty()) printf("Unbalanced\n");
      else if(stack[top]!='['){
        printf("Unbalanced\n");
        break;
      }
      else pop();
    }
    else if(exp[i]==')'){
      if(isEmpty()) printf("Unbalanced\n");
      else if(stack[top]!='('){
        printf("Unbalanced\n");
        break;
      }
      else pop();
    }
    else if(exp[i]=='}'){
      if(isEmpty()) printf("Unbalanced\n");
      else if(stack[top]!='{'){
        printf("Unbalanced\n");
        break;
      }
      else pop();
    }
  }
  if(isEmpty()) printf("Expression is balanced!!");
  return 0;
}
```

Output

```
/tmp/JWsbDeza1l.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.

```c
#include <stdio.h>
#include <stdbool.h>
#define size 7
int front=-1,rear=-1;
int queue[size];
bool isFull(){
  if(rear==size-1) return true;
  return false;
}
bool isEmpty(){
  if (front==-1&&rear==-1) return true;
  return false;
}
void enqueue(){
  if(isFull()){
    printf("Queue overflow\n");
    return;
  }
  int ele;
  printf("Enter the element: ");
  scanf("%d",&ele);
  if(isEmpty()){
    front++;
    rear++;
    queue[front]=ele;
  }
  else queue[++rear]=ele;
  return;
}
void dequeue(){
  if(isEmpty()){
```

```
    printf("Queue underflow\n");
    return;
  }
  int ele;
  if(front==rear){
    ele=queue[rear];
    front=rear=-1;
  }
  else ele=queue[front++];
  printf("Deleted element is %d\n",ele);
  return;
}
void display(){
  if(isEmpty()){
    printf("Queue is empty\n");
    return;
  }
  else{
    printf("The elements are: ");
    for(int i=front;i<=rear;i++) printf("%d\t",queue[i]);
  }
  printf("\n");
  return;
}
int main(void) {
  printf("Enter 1 for Enqueue operation\n");
  printf("Enter 2 for Dequeue operation\n");
  printf("Enter 3 for Display operation\n");
  char c='y';
  int choice;
  while(c=='y'){
    printf("\nEnter the choice: ");
    scanf(" %d",&choice);
    switch(choice){
      case 1:
        enqueue();
        break;
      case 2:
        dequeue();
        break;
      case 3:
        display();
        break;
      default:
        printf("Invalid choice!!\n");
        break;
    }
    printf("Want to continue(y/n): ");
    scanf(" %c",&c);
  }
  printf("\nPrograms ends!!!");
  return 0;
}
```

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.

```c
#include <stdio.h>
int main(void) {
  int rows,columns,size=0;
  printf("Enter the no. of rows: ");
  scanf("%d",&rows);
  printf("Enter the no of columns: ");
  scanf("%d",&columns);
  printf("Enter the elements: \n");
  int sparsematrix[rows][columns];
  for(int i=0;i<rows;i++){
    for(int j=0;j<columns;j++){
      scanf("%d",&sparsematrix[i][j]);
      if(sparsematrix[i][j]!=0) size++;
    }
  }
```

```
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 00 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.

```c
#include <stdio.h>
#define n 5
int arr[n];
int front=-1,rear=-1;
void enqueue(int num){
  if (front==-1&&rear==-1){
    front++;
    rear++;
    arr[rear]=num;
  }
  else if((rear+1)%n==front) printf("Queue overflow\n");
  else{
    rear=(rear+1)%n;
    arr[rear]=num;
  }
}
void dequeue(){
  if(front==-1&&rear==-1) printf("Queue underflow\n");
  else if(front==rear){
    front=-1;
    rear=-1;
  }
  else front=(front+1)%n;
}
void display(){
  if(front==-1 && rear==-1) printf("Queue is empty");
  else{
    int i=front;
    while(i!=rear){
      printf("%d\t",arr[i]);
      i=(i+1)%n;
    }
    printf("%d\t",arr[i]);
  }
  printf("\n");
}
void peek(){
  if(front==-1&&rear==-1) printf("Queue is empty");
  else printf("%d",arr[front]);
  printf("\n");
}
int main(void) {
  printf("Welcome\n");
  char choice='y';
  int select,ele;
```

```
  printf("Enter 1 for enqueue\n");
  printf("Enter 2 for dequeue\n");
  printf("Enter 3 for Display\n");
  printf("Enter 4 for peek\n");
  while(choice=='y'){
    printf("\nEnter the choice: ");
    scanf("%d",&select);
    switch(select){
      case 1:
        printf("Enter the element: ");
        scanf("%d",&ele);
        enqueue(ele);
        break;
      case 2:
        dequeue();
        break;
      case 3:
        display();
        break;
      case 4:
        peek();
        break;
      default:
        printf("Wrong choice entered!!\n");
    }
    printf("Want to continue(y/n): ");
    scanf(" %c",&choice);
  }
  return 0;
}
```

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.

```c
#include <stdio.h>
int swap(int *a,int *b){
  int temp=*a;
  *a=*b;
  *b=temp;
}
int main(void) {
  int n;
  printf("Enter the size: ");
  scanf("%d",&n);
  int arr[n];
  printf("Enter the array: ");
  for(int i=0;i<n;i++){
    scanf("%d",&arr[i]);
  }
  int j=n;
  while(j>=0){
    for(int i=0;i<j-1;i++){
      if(arr[i]>arr[i+1]) swap(&arr[i],&arr[i+1]);
    }
    j--;
  }
  printf("Sorted array is: ");
  for(int i=0;i<n;i++){
    printf("%d\t",arr[i]);
  }
  return 0;
}
```

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.

```c
#include<stdio.h>
void swap(int *a,int*b){
  int temp=*a;
  *a=*b;
  *b=temp;
}
int main(void) {
  int n;
  printf("Enter the number of elements: ");
  scanf("%d",&n);
  int arr[n];
  printf("Enter the elements: ");
  for(int i=0;i<n;i++) scanf("%d",&arr[i]);
  for(int i=1;i<n;i++){
    for(int j=i;j>0;j--){
      if(arr[j-1]<arr[j]) break;
      else swap(&arr[j],&arr[j-1]);
      //if(arr[j-1]>arr[j]) swap(&arr[j],&arr[j-1]); I have replaced this line
          with above two lines of code
    }
  }
  printf("Element after sorting: \n");
  for(int i=0;i<n;i++){
    printf("%d\t",arr[i]);
  }
  return 0;
}
```

Output

```
/tmp/goT37ll4nW.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.

```c
#include <stdio.h>
void merge(int arr[], int s, int mid, int e) {
  int n1 = mid - s + 1, n2 = e - mid;
  int L[n1], R[n2];
  for(int i = 0; i < n1; i++) L[i] = arr[s + i];
  for(int i = 0; i < n2; i++) R[i] = arr[mid + i + 1];
  int i = 0, j = 0, k = s;
  while(i < n1 && j < n2) {
    if(L[i] < R[j]) {
      arr[k] = L[i];
      i++;
    }
    else{
      arr[k] = R[j];
      j++;
    }
    k++;
  }
  while(i < n1) {
    arr[k] = L[i];
    i++;
    k++;
  }
  while(j < n2) {
    arr[k] = R[j];
    j++;
    k++;
  }
  return;
}
void mergesort(int arr[], int s, int e) {
  if (s >= e) return;
  else{
  int mid = s + (e - s) / 2;
  mergesort(arr, s, mid);
  mergesort(arr, mid + 1, e);
  merge(arr, s, mid, e);
  }
}
int main(void) {
  int size;
  printf("Enter the size: ");
  scanf("%d", &size);
  int arr[size];
  printf("Enter the elements: ");
```

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

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.

```
#include <stdio.h>
void swap(int *f,int*s){
    int temp=*f;
    *f=*s;
    *s=temp;
}
int partion(int arr[],int s,int e){
    int i=s,j=e;
    int pivot=s;
    while(i<j){
        while(arr[i]<=arr[pivot]&&i<e) i++;
        while(arr[j]>arr[pivot]&&j>s) j--;
        if(i<j) swap(&arr[i],&arr[j]);
    }
    swap(&arr[pivot],&arr[j]);
    return j;
}
void quicksort(int arr[],int s,int e){
    if(s<e){
        int p=partion(arr,s,e);
        quicksort(arr,s,p-1);
        quicksort(arr,p+1,e);
    }
}
int main(){
    int size;
    printf("Enter the size of the array: ");
    scanf("%d",&size);
    int arr[size];
    printf("Enter the elements: ");
    for(int i=0;i<size;i++) scanf("%d",&arr[i]);
```

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

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.

```c
#include <stdio.h>
void swap(int *a,int*b){
  int temp=*a;
  *a=*b;
  *b=temp;
}
int main(void) {
  int n;
  printf("Enter size of array: ");
  scanf("%d",&n);
  int arr[n];
  printf("Enter elements: ");
  for(int i=0;i<n;i++) scanf("%d",&arr[i]);
  for(int i=0;i<n;i++){
    int min=i;
    for(int j=i;j<n;j++){
      if(arr[min]>arr[j]) min =j;
    }
    swap(&arr[min],&arr[i]);
  }
  for(int i=0;i<n;i++) printf("%d\t",arr[i]);
  return 0;
}
```

Output

```
/tmp/IOobORT5zi.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

```c
#include <stdio.h>
#include<stdlib.h>
struct node{
  int data;
  struct node*next,*prev;
};
void InsertAtHead(struct node**head){
  int d;
  printf("Enter the data: ");
  scanf("%d",&d);
  struct node* temp;
  temp=(struct node*)(malloc(sizeof(struct node)));
  temp->data=d;
  temp->prev=NULL;
  if(*head==NULL){
    temp->next=NULL;
    *head=temp;
  }
  else{
    temp->next=(*head);
    *head=temp;
  }
  return;
}
void InsertAtTail(struct node**head){
  if(*head==NULL){
    InsertAtHead(head);
    return;
  }
  struct node* newnode;
  struct node* temp;
```

```
    newnode=(struct node*)(malloc(sizeof(struct node)));
    int element;
    printf("Enter the element: ");
    scanf("%d",&element);
    newnode->data=element;
    newnode->next=NULL;
    temp=*head;
    while(temp->next!=NULL){
      temp=temp->next;
    }
    temp->next=newnode;
    newnode->prev=temp;
    return;
}
void deletion(struct node**head){
    if(*head==NULL){
      printf("List is empty\n");
      return;
    }
    int element;
    printf("Enter the element to delete: ");
    scanf("%d",&element);
    struct node*prev=NULL,*curr=*head;
    if(curr->data==element){//Handling first element
      *head=(*head)->next;
      free(curr);
      return;
    }
    while(curr!=NULL&&curr->data!=element){
      prev=curr;
      curr=curr->next;
    }
    if(curr==NULL){
      printf("Element not found!!!\n");
      return;
    }
    prev->next=curr->next;
    curr->next->prev=prev;
    free(curr);
    return;
}
void display(struct node *head){
    if(head==NULL){
      printf("Linked list has no elements!!!\n");
      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 }
```

```c
void dequeue(){
  struct node* temp=front;
  if(front==NULL &&rear==NULL){
    printf("Queue underflow\n");
    return;
  }
  else if(front==rear){
    front=rear=NULL;
  }
  else{
    front=front->next;
  }
  free(temp);
}
void display(){
  if(front==NULL && rear==NULL) printf("Queue is empty");
  else{
    struct node* temp=front;
    while(temp!=NULL){
      printf("%d\t",temp->data);
      temp=temp->next;
    }
  }
  printf("\n");
}
void peek(){
  if (front==NULL&& rear==NULL) printf("Queue is empty\n");
  else printf("%d\n",front->data);
}
int main(void) {
  char choice='y';
  printf("Enter 1 for enqueue\n");
  printf("Enter 2 for dequeue\n");
  printf("Enter 3 for display\n");
  printf("Enter 4 for peek\n");
  while(choice=='y'){
    int select;
    printf("\nEnter the choice: ");
    scanf("%d",&select);
    switch(select){
      case 1:
        enqueue();
        break;
      case 2:
        dequeue();
        break;
      case 3:
        display();
        break;
      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/4S6I4OJdKE.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

```c
#include <stdio.h>
#include<stdlib.h>
struct node{
  int data;
  struct node*next;
};
void InsertAtHead(struct node**head){
  int d;
  printf("Enter the data: ");
  scanf("%d",&d);
  struct node* temp;
  temp=(struct node*)(malloc(sizeof(struct node)));
  temp->data=d;
  if(*head==NULL){
    temp->next=NULL;
    *head=temp;
  }
  else{
    temp->next=(*head);
    *head=temp;
  }
  return;
}
void InsertAtTail(struct node**head){
  if(*head==NULL){
    InsertAtHead(head);
    return;
  }
  struct node* newnode;
  struct node* temp;
  newnode=(struct node*)(malloc(sizeof(struct node)));
   int element;
   printf("Enter the element: ");
   scanf("%d",&element);
   newnode->data=element;
   newnode->next=NULL;
   temp=*head;
   while(temp->next!=NULL){
     temp=temp->next;
   }
   temp->next=newnode;
   return;
}
void InsertAtPos(struct node **head){
  if(*head==NULL){
    InsertAtHead(head);
    return;
  }
  struct node *temp=*head;
  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/yIJaQuYRzL.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.

```c
#include <stdio.h>
#include<stdlib.h>
struct node{
   int data;
   struct node* next;
};
struct node* insert(struct node** top){
   struct node* newnode;
   newnode=(struct node*)malloc(sizeof(struct node));
   int element;
   printf("Enter the element: ");
   scanf("%d",&element);
   newnode->data=element;
   newnode->next=*top;
   *top=newnode;
   return *top;
}
struct node* deletion(struct node** top){
   if (*top==NULL){
     printf("Stack underflow!!!\n");
```

```c
    return *top;
  }
  struct node* newnode;
  newnode=*top;
  *top=(*top)->next;
  free(newnode);
  return *top;
}
void display(struct node* top){
  if(top==NULL){
    printf("Stack is empty!!!\n");
    return;
  }
  printf("Elements are: ");
  while(top!=NULL){
    printf("%d\t",top->data);
    top=top->next;
  }
  printf("\n");
  return;
}
int main(void) {
  struct node *top=NULL;
  char ch='y';
  int choice;
  printf("Enter 1 for insertion\n");
  printf("Enter 2 for deletion\n");
  printf("Enter 3 for traversal\n");
  while(ch=='y'){
    printf("Enter the choice: ");
    scanf("%d",&choice);
    switch(choice){
      case 1:
        insert(&top);
        break;
      case 2:
        deletion(&top);
        break;
      case 3:
        display(top);
        break;
      default:
        printf("Invalid choice entered!!!\n");
    }
    printf("\nWant to continue(y/n): ");
    scanf(" %c",&ch);
  }
  printf("Program executed succesfully!!!");
  return 0;
}
```

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.

```c
#include <stdio.h>
#include<time.h>
int binarysearch(int ele,int n,int *arr){
  int s=0,e=n-1;
  int mid=s+(e-s)/2;
  while(s<=e){
    if(arr[mid]==ele) return mid;
    else if(arr[mid]>ele) e=mid-1;
    else if(arr[mid]<ele) s=mid+1;
    mid=s+(e-s)/2;
  }
  return -1;
}
int main(void) {
  int n;
  printf("Enter the number of elements: ");
  scanf("%d",&n);
  int arr[n];
  printf("Enter the sorted element: ");
  for(int i=0;i<n;i++){
    scanf("%d",&arr[i]);
  }
  time_t start,end;
  start=time(NULL);
  int ele,index;
  printf("Enter the element to search: ");
  scanf("%d",&ele);
  index=binarysearch(ele,n,arr);
  if(index==-1) printf("Element not found!!\n");
  else printf("Element occur at index %d\n",index);
  end=time(NULL);
  printf("Execution time for binary search is : %f seconds",difftime(end,start
      ));
  return 0;
}
```

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.

```c
#include <stdio.h>
#include<time.h>
int linearsearch(int ele,int n,int *arr){
  for(int i=0;i<n;i++){
    if(arr[i]==ele) return i;
  }
  return -1;
}
int main(void) {
  int n;
  printf("Enter the number of elements: ");
  scanf("%d",&n);
  int arr[n];
  printf("Enter the element: ");
  for(int i=0;i<n;i++){
    scanf("%d",&arr[i]);
  }
  time_t start,end;
  start=time(NULL);
  int ele;
  printf("Enter the element to search: ");
  scanf("%d",&ele);
  int index=linearsearch(ele,n,arr);
  if(index==-1) printf("Element not found!!\n");
  else printf("Element occur at index %d\n",index);
  end=time(NULL);
  printf("Execution time for linear search is : %f seconds\n",difftime(end,start
      ));
  return 0;
}
```

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).

```c
#include <stdio.h>
#include<stdlib.h>
struct node{
  int data;
  struct node* left,*right;
};
struct node* createnode(int val){
  struct node* root=(struct node*)malloc(sizeof(struct node));
  root->data=val;
  root->left=NULL;
  root->right=NULL;
  return root;
}
void inorder(struct node *root){
  if(root!=NULL){
    inorder(root->left);
    printf("%d\t",root->data);
    inorder(root->right);
  }
}
void preorder(struct node *root){
  if(root!=NULL){
    printf("%d\t",root->data);
    preorder(root->left);
    preorder(root->right);
  }
}
void postorder(struct node *root){
  if(root!=NULL){
    postorder(root->left);
    postorder(root->right);
    printf("%d\t",root->data);
  }
}
void insert(struct node**root,int ele){
  if((*root)==NULL){
    (*root)=createnode(ele);
    return;
  }
  if((*root)->data<ele) insert(&((*root)->right),ele);;
  if((*root)->data>ele) insert(&((*root)->left),ele);
}
struct node* nextinorder(struct node**root){
  struct node*curr=*root;
  while(curr&&curr->left!=NULL)
```

```
    curr=curr->left;
  return curr;
}
void delete(struct node**root,int ele){
  if((*root)==NULL) printf("\nElement not found!!\n");
  else if((*root)->data<ele) delete(&((*root)->right),ele);
  else if((*root)->data>ele) delete(&((*root)->left),ele);
  else if((*root)->left&&(*root)->right){
    struct node* next=nextinorder(&((*root)->right));
    (*root)->data=next->data;
    delete(&((*root)->right),next->data);
  }
  else{
    struct node**temp=root;
    if((*root)->left==NULL&&(*root)->right==NULL) (*root)=NULL;
    else if((*root)->left!=NULL) (*root)=(*root)->left;
    else if((*root)->right!=NULL) (*root)=(*root)->right;
    free(*temp);
  }
}
int main(void) {
  printf("Welcome\n");
  char ch='y';
  int ele;
  struct node *root=NULL;
  printf("Enter 1 for insertion\n");
  printf("Enter 2 for deletion\n");
  printf("Enter 3 for inorder traversal\n");
  printf("Enter 4 for preorder traversal\n");
  printf("Enter 5 for postorderorder traversal\n");
  while(ch=='y'){
    int choice;
    printf("Enter the choice: ");
    scanf("%d",&choice);
    switch(choice){
      case 1:
        printf("Enter element to insert: ");
        scanf("%d",&ele);
        insert(&root,ele);
        break;
      case 2:
        printf("Enter element to delete: ");
        scanf("%d",&ele);
        delete(&root,ele);
        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 ===
```