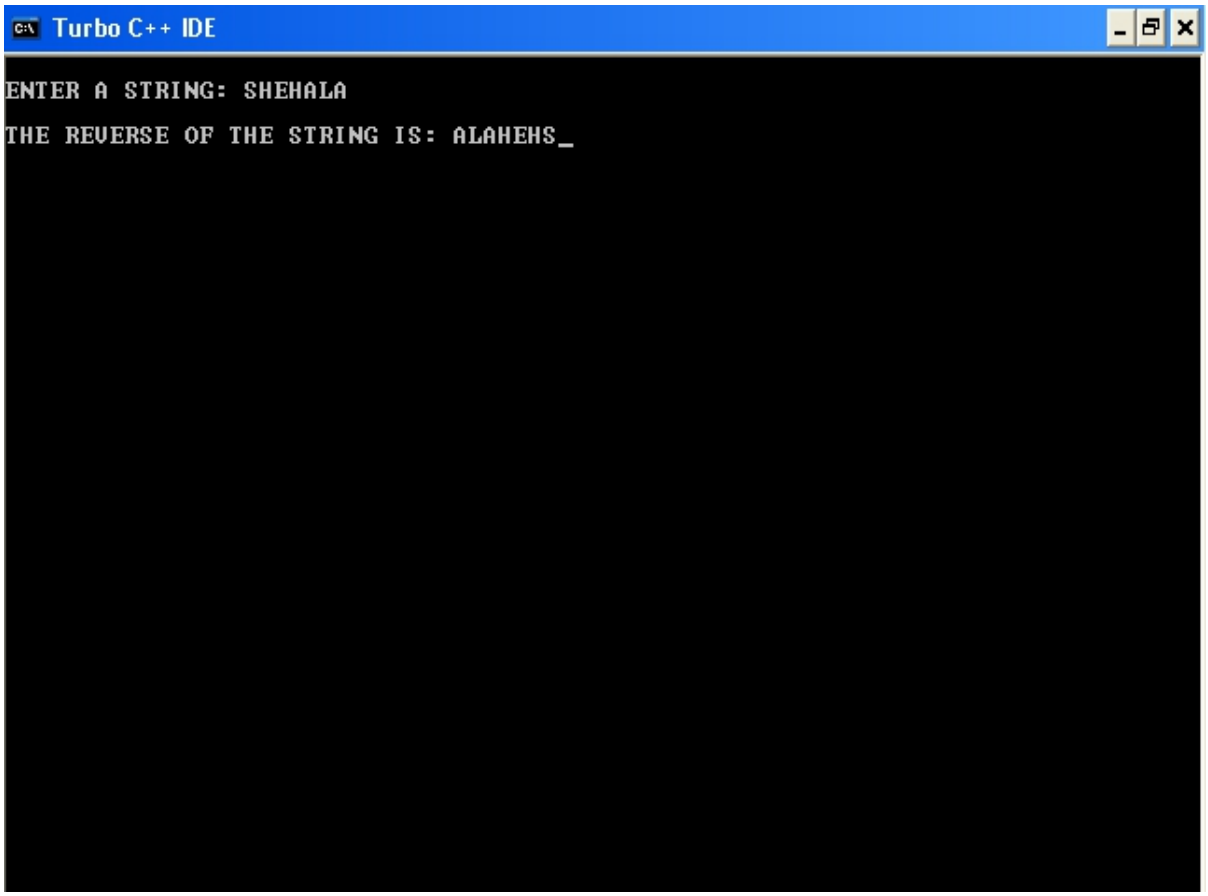
1.Reverse a string using pointers.

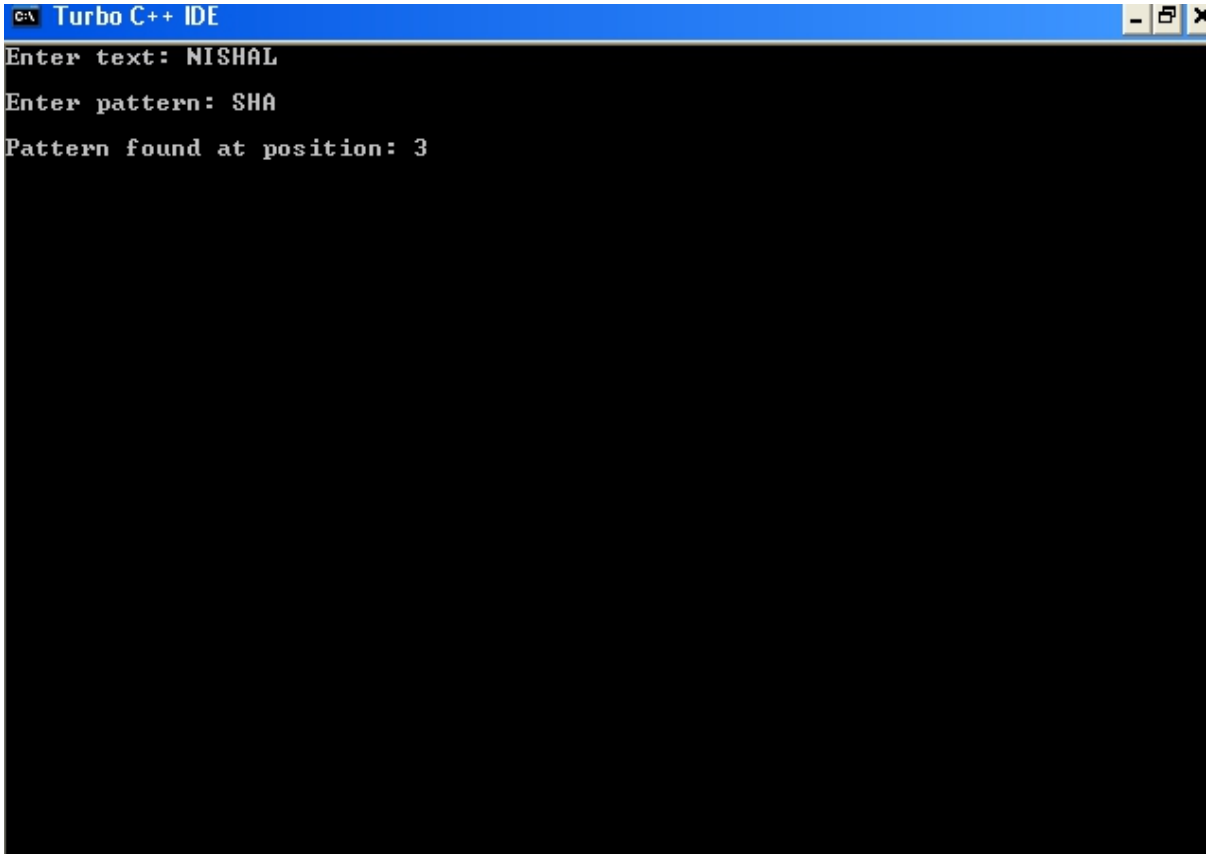
```
#include <stdio.h>
#include <conio.h>
#include<string.h>
void main()
{
        char *s;
        int  len,i;
        clrscr();
        printf("\nENTER A STRING: ");
        gets(s);
        len=strlen(s);
        printf("\nTHE REVERSE OF THE STRING IS:");
        for(i=len;i>=0;i--)
        printf("%c",*(s+i));
        getch();
}
```



2.Implement Pattern matching algorithm.

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
void main()
{
inti,j,k,n,m,flag=0;
char t[40],p[30];
clrscr();
printf("Enter text: ");
gets(t);
printf("\nEnter pattern: ");
gets(p);
 n=strlen(t);
 m=strlen(p);
for(i=0;i<=n-m;i++)
 {
  j=0;
while(j<m && p[j]==t[j+i])
   {
j++;
if(j==m)
     {
flag=1;
      k=i+1;
```

```
    }
else
flag=0;
  }
 }
if(flag==1)
printf("\nPattern found at position: %d\n ",k);
else
printf("\nPattern not found in text \n");
getch();
}
```



3.Append 2 arrays

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a[5],b[5],c[10],i,j,n,m ;
clrscr();
printf("Enter the limit of the first array");
scanf("%d",&n);
printf("Enter the elements of first array");
for(i=0;i<n;i++)
scanf("%d",&a[i]);
printf("Enter the limit of the second array");
scanf("%d",&m);
printf("Enter the elements of second array");
for (i=0;i<m;i++)
```

```
{
scanf("%d",&b[i]);
}
for (i=0;i<n;i++)
c[i]=a[i];
for(j=0;j<m;j++)
c[i++]=b[j];
j=i;
printf("After concatenation:\n");
for (i=0; i<j; i++)
{
printf("%d\n", c[i]);
}
getch ( );
}
```

```
Enter the limit of the first array:2
Enter the elements of first array
6 7
Enter the limit of the second array:3
Enter the elements of second array
8 9 10
After concatenation:
6
7
8
9
10
_
```

4.Search an element in the 2 dimentional array

```
#include<stdio.h>

#include<conio.h>

void main()

{
```

```
int a[10][10],i, j, row, col, val,found=0;
clrscr();
printf(" Enter the number of row: ");
scanf("%d", &row);
printf(" Enter the number of columns: ");
scanf("%d", &col);
printf(" Enter the Matrix elements:");
for(i=0; i<row; i++)
 {
  for(j=0; j<col; j++)
  {
   scanf("%d", &a[i][j]);
  }
 }
printf(" \nThe Entered Matrix is:\n");
for(i=0; i<row; i++)
 {
  for(j=0; j<col; j++)
  {
   printf("%d \t", a[i][j]);
  }
  printf("\n");
 }
printf(" Enetr the element to be searched: ");
scanf("%d", &val);
for(i=0; i<row; i++)
 {
  for(j=0; j<col; j++)
```

```
  {
    if(a[i][j]==val)
     {
      printf("Item found at row: %d  and column :%d ", i+1,j+1);
      found=1;
     }
  }
 }
 if(found==0)
 {
  printf("The item was not in the list:");
 }
 getch();
}
```

Turbo C++ IDE

```
 Enter the number of row: 4
 Enter the number of columns: 2
 Enter the Matrix elements:11
22
33
44
55
66
77
88

The Entered Matrix is:
11      22
33      44
55      66
77      88
 Enetr the element to be searched: 99
The item was not in the list:
```

Turbo C++ IDE

```
 Enter the number of row: 2
 Enter the number of columns: 3
 Enter the Matrix elements:12
21
23
32
43
34

The Entered Matrix is:
12      21      23
32      43      34
 Enetr the element to be searched: 34
Item found at row: 2  and column :3 _
```

5.Search an element in the array using binary search

```
#include <stdio.h>
#include<conio.h>
int main()
{
     int c, first, last, middle, n, search, array[100];
    Clrscr();
    printf("Enter number of elements\n");
   scanf("%d",&n);
   printf("Enter %d integers\n", n);
  for (c = 0; c < n; c++)
   scanf("%d",&array[c]);
 printf("Enter value to find\n");
 scanf("%d", &search);
 first = 0;
 last = n - 1;
 middle = (first+last)/2;
 while (first <= last)
 {
   if (array[middle] < search)
  first = middle + 1;
  else if (array[middle] == search)
   {
    printf("%d found at location %d.\n", search, middle+1);
    break;
```

```
  }
else
  last = middle - 1;
  middle = (first + last)/2;
 }
if (first > last)
printf("Not found! %d isn't present in the list.\n", search);
return 0;
}
```

```
Enter number of elements
4
Enter 4 integers
22 33 44 55
Enter value to find
55
55 found at location 4.
_
```

6. Read Sparse matrix and display its triplet representation using array

```
#include<stdio.h>
#include<conio.h>
#define MAX 10
```

```
void main()
{
int a[10][10],b[MAX][3],i,j,k,r,c;
clrscr();
printf("Enter the order of  matrix");
scanf("%d%d",&r,&c);
printf("Enter the matrix");
for(i=0;i<r;i++)
{
for(j=0;j<c;j++)
{
scanf("%d",&a[i][j]);
k=1;
b[0][0]=r;
b[0][1]=c;
}
}
for(i=0;i<r;i++)
{
for(j=0;j<c;j++)
{
if(a[i][j]!=0)
{
b[k][0]=i;
```

```
b[k][1]=j;
b[k][2]=a[i][j];
k++;
}
}
b[0][2]=k-1;
}
printf("triplet representation\n");
c=b[0][2];
for(i=0;i<=c;i++)
printf("%d\t%d\t%d\n",b[i][0],b[i][1],b[i][2]);
getch();
}
```

Output

```
Enter the order of  matrix
3 3
Enter the matrix
1 1 0
0 0 1
1 0 0
triplet representation
3       3       4
0       0       1
0       1       1
1       2       1
2       0       1
```

7. Implement stack using array

```
#include<stdio.h>
```

```
int stack[100],choice,n,top,x,i;
void push(void);
void pop(void);
void display(void);
int main()
{
   //clrscr();
top=-1;
printf("\n Enter the size of STACK[MAX=100]:");
scanf("%d",&n);
printf("\n\t STACK OPERATIONS USING ARRAY");
printf("\n\t--------------------------------");
printf("\n\t 1.PUSH\n\t 2.POP\n\t 3.DISPLAY\n\t 4.EXIT");
do
   {
printf("\n Enter the Choice:");
scanf("%d",&choice);
switch(choice)
       {
case 1:
           {
push();
break;
           }
case 2:
```

```
        {
pop();
break;
        }
case 3:
        {
display();
break;
        }
case 4:
        {
printf("\n\t EXIT POINT ");
break;
        }
default:
        {
printf ("\n\t Please Enter a Valid Choice(1/2/3/4)");
        }


      }
   }
while(choice!=4);
return 0;
}
```

```
be pushed:");
scanf("%d",&x);
top++;
stack[top]=x;
   }
}
void pop()
{
if(top<=-1)
   {
printf("\n\t Stack is under flow");
   }
else
   {
printf("\n\t The popped elements is %d",stack[top]);
top--;
   }
}
void display()
{
if(top>=0)
   {
printf("\n The elements in STACK \n");
for(i=top; i>=0; i--)
printf("\n%d",stack[i]);
```

```
printf("\n Press Next Choice");
   }
else
   {
printf("\n The STACK is empty");
   }

}
```

Output

```
Turbo C++ IDE

Enter the size of STACK[MAX=100]:2

        STACK OPERATIONS USING ARRAY
       ---------------------------------
        1.PUSH
        2.POP
        3.DISPLAY
        4.EXIT
Enter the Choice:1
Enter a value to be pushed:4

Enter the Choice:1
Enter a value to be pushed:5

Enter the Choice:2

        The popped elements is 5
Enter the Choice:2

        The popped elements is 4
Enter the Choice:3

The STACK is empty
Enter the Choice:1
Enter a value to be pushed:2

Enter the Choice:1
Enter a value to be pushed:4

Enter the Choice:3

The elements in STACK

4
2
Press Next Choice
Enter the Choice:^C
```

8. STACK using LINKEDLIST

```
#include<stdio.h>
#include<conio.h>
#include<malloc.h>
struct stack
{
 int data;
 struct stack *next;
};
struct stack *top=NULL;
struct stack *push(struct stack *, int);
struct stack *display(struct stack *);
struct stack *pop(struct stack *);
int main()
{
 int val, option;
 clrscr();
 do
```

```
{
 printf("\nMAIN MENU");
 printf("\n 1. PUSH");
 printf("\n 2. POP");
 printf("\n 3. DISPLAY");
 printf("\n 4. EXIT");
 printf("\n Enter your option:");
 scanf("%d", &option);
 switch(option)
 {
  case 1:
     printf("\n Enter the number to be pushed on stack:");
     scanf("%d", &val);
     top=push(top,val);
     break;
  case 2:
     top=pop(top);
     break;
  case 3:
     top=display(top);
     break;
  case 4:exit(0);
   default:
   printf("Invalid choice");
  }
```

```
 }while(option !=4);
 getch();
 return 0;
}
struct stack *push(struct stack *top, int val)
{
 struct stack *ptr;
 ptr=(struct stack *)malloc(sizeof(struct stack));
 ptr->data=val;
 if(top==NULL)
 {
  ptr->next=NULL;
  top=ptr;
 }
 else
 {
  ptr->next=top;
  top=ptr;
 }
 return top;
}

struct stack *display(struct stack *top)
{
 struct stack *ptr;
```

```
  ptr=top;
  if(top==NULL)
   printf("\n STACK IS EMPTY");
  else
  {
   while(ptr!=NULL)
   {
    printf("\n%d", ptr->data);
    ptr=ptr->next;
   }
 }
 return top;
}

struct stack *pop(struct stack *top)
{
 struct stack *ptr;
 ptr=top;
 if(top==NULL)
   printf("\n STACK UNDERFLOW");
 else
  {
   top=top->next;
   printf("\n The value being deleted is :%d",ptr->data);
   free(ptr);
```

```
  }
  return top;
 }
```

Output

```
MAIN MENU
 1. PUSH
 2. POP
 3. DISPLAY
 4. EXIT
 Enter your option:1

 Enter the number to be pushed on stack:3

MAIN MENU
 1. PUSH
 2. POP
 3. DISPLAY
 4. EXIT
 Enter your option:1

 Enter the number to be pushed on stack:6

MAIN MENU
 1. PUSH
 2. POP
 3. DISPLAY
 4. EXIT
 Enter your option:3
```

```
6
3
MAIN MENU
 1. PUSH
 2. POP
 3. DISPLAY
 4. EXIT
 Enter your option:2

 The value being deleted is :6
MAIN MENU
 1. PUSH
 2. POP
 3. DISPLAY
 4. EXIT
 Enter your option:3

3
MAIN MENU
 1. PUSH
 2. POP
 3. DISPLAY
 4. EXIT
 Enter your option:4_
```

9. Evaluation Postfix expression

```
#include<stdio.h>
#include<conio.h>
int stack[20];
int top = -1;

void push(int x)
{
	stack[++top] = x;
}

int pop()
{
	return stack[top--];
}
```

```
int main()
{
        char exp[20];
        char *e;
       clrscr();
        int n1,n2,n3,num;
        printf("Enter the expression :: ");
        scanf("%s",exp);
        e = exp;
        while(*e != '\0')
        {
                if(isdigit(*e))
                {
                        num = *e - 48;
                        push(num);
                }
                else
                {
                        n1 = pop();
                        n2 = pop();
                        switch(*e)
                        {
                                case '+':
                                {
                                        n3 = n1 + n2;
                    break;
                                }
                                case '-':
                                {
                                        n3 = n2 - n1;
                                        break;
                                }
                                case '*':
                                {
                                        n3 = n1 * n2;
                                        break;
                                }
                                case '/':
                                {
                                        n3 = n2 / n1;
```

```
                        break;
                    }
                }
                push(n3);
            }
            e++;
        }
        printf("\nThe result of expression %s  =  %d\n\n",exp,pop());
        return 0;

}
```

Output

```
Enter the Postfix expression :: 562+*84/-

The result of expression 562+*84/- = 38
```

10.Implement Queue using array

```
#include <stdio.h>
#include<conio.h>
#define MAX 50
int queue[MAX];
int rear = - 1;
int front = - 1;
```

```
void delete() ;
void insert();
void display();
void main()
{
    int choice;
    while (1)
    {
        printf("1.Insert element to queue \n");
        printf("2.Delete element from queue \n");
        printf("3.Display all elements of queue \n");
        printf("4.Quit \n");
        printf("Enter your choice : ");
        scanf("%d", &choice);
        switch (choice)
        {
            case 1:
            insert();
            break;
            case 2:
            delet();
            break;
            case 3:
            display();
            break;
            case 4:
            exit(1);
            default:
            printf("Wrong choice \n");
        }
    }
}
void insert()
{
    int additem;
    if (rear == MAX - 1)
    printf("Queue Overflow \n");
    else
    {
        if (front == - 1)
```

```
        front = 0;
        printf("Inset the element in queue : ");
        scanf("%d", &additem);
        rear = rear + 1;
        queue[rear] = additem;
    }
}
void delete()
{
    if (front == - 1 || front > rear)
    {
        printf("Queue Underflow \n");
        return ;
    }
    else
    {
        printf("Element deleted from queue is : %d\n", queue[front]);
        front = front + 1;
    }
}
void display()
{
    int i;
    if (front == - 1)
        printf("Queue is empty \n");
    else
    {
        printf("Queue is : \n");
        for (i = front; i <= rear; i++)
            printf("%d ", queue[i]);
        printf("\n");
    }
}
```

Output

```
Inset the element in queue : 5
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Quit
Enter your choice : 1
Inset the element in queue : 8
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Quit
Enter your choice : 1
Inset the element in queue : 9
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Quit
Enter your choice : 3
Queue is :
5 8 9
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Quit
Enter your choice :
```

```
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Quit
Enter your choice : 3
Queue is :
5 8 8 9
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Quit
Enter your choice : 2
Element deleted from queue is : 5
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Quit
Enter your choice : 3
Queue is :
8 8 9
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Quit
Enter your choice : 4_
```

11.Implement Queue using linked list

```
#include<stdio.h>
#include<conio.h>

struct Node
{
   int data;
   struct Node *next;
}*front = NULL,*rear = NULL;

void insert(int);
void delete();
void display();

void main()
{
   int choice, value;
   clrscr();
   printf("\n:: Queue Implementation using Linked List ::\n");
   while(1){
      printf("\n****** MENU ******\n");
      printf("1. Insert\n2. Delete\n3. Display\n4. Exit\n");
      printf("Enter your choice: ");
```

```
    scanf("%d",&choice);
    switch(choice){
        case 1: printf("Enter the value to be insert: ");
                scanf("%d", &value);
                insert(value);
                break;
        case 2: delete(); break;
        case 3: display(); break;
        case 4: exit(0);
        default: printf("\nWrong selection!!! Please try again!!!\n");
    }
  }
}
void insert(int value)
{
  struct Node *newNode;
  newNode = (struct Node*)malloc(sizeof(struct Node));
  newNode->data = value;
  newNode -> next = NULL;
  if(front == NULL)
    front = rear = newNode;
  else{
    rear -> next = newNode;
    rear = newNode;
  }
```

```
  printf("\nInsertion is Success!!!\n");
}
void delete()
{
  if(front == NULL)
    printf("\nQueue is Empty!!!\n");
  else{
    struct Node *temp = front;
    front = front -> next;
    printf("\nDeleted element: %d\n", temp->data);
    free(temp);
  }
}
void display()
{
  if(front == NULL)
    printf("\nQueue is Empty!!!\n");
  else{
    struct Node *temp = front;
    while(temp->next != NULL){
        printf("%d--->",temp->data);
        temp = temp -> next;
    }
    printf("%d--->NULL\n",temp->data);
  }
```

}

Output

```
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Quit
Enter your choice : 1
Inset the element in queue : 3
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Quit
Enter your choice : 1
Inset the element in queue : 9
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Quit
Enter your choice : 3
Queue is :
3 9
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Quit
Enter your choice : _
```

```
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Quit
Enter your choice : 3
Queue is :
3 9
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Quit
Enter your choice : 2
Element deleted from queue is : 3
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Quit
Enter your choice : 3
Queue is :
9
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Quit
Enter your choice : 4
```

12: create a singly linked list of n nodes and diplay it.

```
#include <stdio.h>
#include <stdlib.h>

struct node
{
int num;
struct node *nextptr;
}*stnode;

void createNodeList(int n);
void displayList();
int main()
{
int n;
clrscr();
        printf("\n\n Linked List : To create and display Singly
Linked List :\n");
        printf("-------------------------------------------------------
\n");
```

```
   printf(" Input the number of nodes : ");
   scanf("%d", &n);
   createNodeList(n);
   printf("\n Data entered in the list : \n");
displayList();
getch();
   return 0;
}
void createNodeList(int n)
{
   struct node *fnNode, *tmp;
   int num, i;
   stnode = (struct node *)malloc(sizeof(struct node));

if(stnode == NULL)
   {
      printf(" Memory can not be allocated.");
   }
   else
```

```
{
    printf(" Input data for node 1 : ");
    scanf("%d", &num);
    stnode->num = num;
    stnode->nextptr = NULL; // links the address field to NULL
    tmp = stnode;
    for(i=2; i<=n; i++)
    {
        fnNode = (struct node *)malloc(sizeof(struct node));
        if(fnNode == NULL)
        {
            printf(" Memory can not be allocated.");
            break;
        }
        else
        {
            printf(" Input data for node %d : ", i);
            scanf(" %d", &num);

            fnNode->num = num;
```

```
            fnNode->nextptr = NULL;
            tmp->nextptr = fnNode;
            tmp = tmp->nextptr;
        }
    }
    }
}
void displayList()
{
    struct node *tmp;
    if(stnode == NULL)
    {
        printf(" List is empty.");
    }
    else
    {
        tmp = stnode;
        while(tmp != NULL)
        {
            printf(" Data = %d\n", tmp->num);
```

```
        tmp = tmp->nextptr
}
  }
  }
```

Output

```
 Linked List : To create and display Singly Linked List :
-------------------------------------------------------------
 Input the number of nodes : 3
 Input data for node 1 : 6
 Input data for node 2 : 7
 Input data for node 3 : 8

 Data entered in the list :
 Data = 6
 Data = 7
 Data = 8
-
```

13. Delete a given node from a singly linked list?

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<math.h>
```

```
struct node
{
 int data;
 struct node *next;
};
struct node *start;

void insertbeg(void)
{
 struct node *nn;
 int a;
  nn=(struct node *)malloc(sizeof(struct node));
 printf("enter data:");
 scanf("%d",&nn->data);
 a=nn->data;
 if(start==NULL)
 {
 nn->next=NULL;
 start=nn;
 }
```

```
 else
 {
 nn->next=start;
 start=nn;
 }
 printf("%d succ. inserted\n",a);
 return;
}

void deletion(void)
{
 struct node *pt,*t;
 int x;
 if(start==NULL)
 {
 printf("sll is empty\n");
 return;
 }
 printf("enter data to be deleted:");
 scanf("%d",&x);
```

```
if(x==start->data)
{
t=start;



start=start->next;
free(t);
printf("%d is succ. deleted\n",x);
return;
}
pt=start;
t=start->next;
while(t!=NULL&&t->data!=x)
{
pt=t;t=t->next;
}
if(t==NULL)
{
printf("%d does not exist\n",x);return;
}
```

```
 else
 {
 pt->next=t->next;
 }
 printf("%d is succ. deleted\n",x);
 free(t);
 return;
}
void display(void)
{
 struct node *temp;
 if(start==NULL)
 {
 printf("sll is empty\n");
 return;
 }
 printf("elements are:\n");
 temp=start;
 while(temp!=NULL)
 {
```

```
 printf("%d\n",temp->data);
 temp=temp->next;
 }
 return;
}

int main()
{
 int c,a; start=NULL;
 clrscr();
 do
 {
 printf("1:insert\n2:delete\n3:display\n4:exit\nenter choice:");
 scanf("%d",&c);
 switch(c)
 {
 case 1:insertbeg(); break;
 case 2:deletion(); break;
 case 3:display(); break;
 case 4:printf("program ends\n");break;
```

```
 default:printf("wrong choice\n");
 break;
 }
 }while(c!=4);return 0;
}
```

```
    2:delete
    3:display
    4:exit
    enter choice:1
    enter data:5
    5 succ. inserted
    1:insert
    2:delete
    3:display
    4:exit
    enter choice:1
    enter data:7
    7 succ. inserted
    1:insert
    2:delete
    3:display
    4:exit
    enter choice:1
    enter data:8
    8 succ. inserted
    1:insert
    2:delete
    3:display
    4:exit
    enter choice:3
```

```
enter choice:3
elements are:
8
7
5
1:insert
2:delete
3:display
4:exit
enter choice:2
enter data to be deleted:7
7 is succ. deleted
1:insert
2:delete
3:display
4:exit
enter choice:3
elements are:
8
5
1:insert
2:delete
3:display
4:exit
enter choice:4_
```

14. Create a doubly linked list of integers and display in forward and backward direction.

```
#include <stdio.h>

#include <stdlib.h>


struct node* create(struct node *, struct node **, int);

void display(struct node*);

void displays(struct node*, struct node*);



struct node

{

int data;

struct node *rptr, *lptr;

};
```

```
int main()
{
int i, n, value;

 struct node *head, *tail;
 head= NULL;
 tail= NULL;
 clrscr();
 printf("\nEnter the number of values u want to enter\n");
 scanf("%d", &n);
 printf("\nEnter the number you want to enter\n");
 for(i=0; i<n; i++)
 {
 scanf("%d",&value);
 head=create(head, &tail, value);
 }

 printf("\nThe data in forward direction is printed below\n");
 display(head);

 printf("\nThe data in backward direction is printed below\n");
 displays(tail, head);
 getch();
```

```
 return 0;
}

struct node* create(struct node *head1, struct node **tail1, int dat)
{
 struct node* newnode, *temp;
 newnode= (struct node*) malloc (sizeof(struct node));
 newnode->data=dat;
 newnode->rptr= newnode->lptr= NULL;

 if(head1 == NULL)
 {
  newnode->lptr=newnode->rptr=NULL;
  head1=newnode;
 }

   temp=head1;
 while(temp->rptr != NULL)
 temp=temp->rptr;

  temp->rptr= newnode;
  newnode->lptr=temp;
  newnode->rptr=NULL;
  *tail1 = newnode;
  temp=temp->rptr;
```

```
return head1;

}

void display(struct node* head)

{
 while(head!= NULL)
  {
   printf("%d\n",head->data);
   head=head->rptr;
  }
}

void displays(struct node *tail, struct node *head)
{
  while (tail != head)
  {
   printf("%d\n", tail->data);
   tail=tail->lptr;
  }
```

```
  if(tail == head)
   printf("%d\n", tail->data);
}
```

Output

```
Enter the number of values u want to enter
3

Enter the number you want to enter
7 8 9

The data in forward direction is printed below
7
8
9

The data in backward direction is printed below
9
8
7
-
```

15.implentation insertion sort

```
#include<stdio.h>
int main()
{
  int i, j, count, temp, number[25];
  clrscr();
  printf("Enter the limit: ");
  scanf("%d",&count);
  printf("Enter %d elements: ", count);
   for(i=0;i<count;i++)
    scanf("%d",&number[i]);
  for(i=1;i<count;i++){
    temp=number[i];
    j=i-1;
    while((temp<number[j])&&(j>=0)){
      number[j+1]=number[j];
      j=j-1;
    }
    number[j+1]=temp;
  }
  printf(" Sorted elements: ");
  for(i=0;i<count;i++)
  printf(" %d",number[i]);
```

```
  getch();
  return 0;
}
```

Output

```
Enter the limit: 4
Enter 4 elements: 3 6 5 2
 Sorted elements:  2 3 5 6_
```

15. implementation selection sort

```
#include <stdio.h>
#include<conio.h>
int main()
{
int a[100], n, i, j, position, swap;
```

```
clrscr();
printf("Enter number of elements\n");
scanf("%d", &n);
printf("Enter %d Numbers\n", n);
for (i = 0; i < n; i++)
scanf("%d", &a[i]);
for(i = 0; i < n - 1; i++)
{
position=i;
for(j = i + 1; j < n; j++)
{
if(a[position] > a[j])
position=j;
}
if(position != i)
{
swap=a[i];
a[i]=a[position];
a[position]=swap;
}
}
printf("Sorted Array: ");
for(i = 0; i < n; i++)
printf("%d ", a[i]);
```

```
getch();
return 0;
}
```

Output

```
Enter number of elements
4
Enter 4 Numbers
5 3 8 1
Sorted Array: 1 3 5 8
```

17.Implement exchange sort

```
#include<stdio.h>
#include<conio.h>

int main()
{
 int array[10],n,i,j,temp;
 clrscr();
 printf("Enter the limit:");
 scanf("%d",&n);
 printf("Enter the  numbers\n");
 for (i = 0; i < n; i++)
        {

                scanf("%d",&array[i]);
        }

        for(i = 0; i < (n -1); i++)
        {
                for (j=(i + 1); j < n; j++)
                {
                        if (array[i] > array[j])
                        {
                                temp = array[i];
                                array[i] = array[j];
                                array[j] = temp;
                        }
                }
        }
        printf("sorted elements :");
        for (i = 0; i < n; i++)
        {
                printf("%d ", array[i]);
```

```
        }
getch();
return 0;
}
```

Output

```
Enter the limit:4
Enter the  numbers
7 3 5 1
sorted elements :1 3 5 7
```

18. search an element in a binary search tree ?

```
#include <stdio.h>

#include <stdlib.h>

#include<conio.h>

struct TreeNode

{

   int data;
```

```
    struct TreeNode *leftChildNode;
    struct TreeNode *rightChildNode;
};
typedef struct TreeNode node;
node *rootNode = NULL;
 void insertNode(int i, node **n)
 {
    if (*n == NULL)
  {
            (*n) = (node*)malloc(sizeof(node));
        (*n)->leftChildNode = NULL;
        (*n)->rightChildNode = NULL;
        (*n)->data = i;
    }
    else if ((*n)->data == i)
        printf("\nThis value already exists in the tree!");
    else if (i > (*n)->data)
        insertNode(i, &((*n)->rightChildNode));
    else
        insertNode(i, &((*n)->leftChildNode));
}
```

```
void searchNode(int i, node **n)
{
   if (*n == NULL)
      printf("\nValue does not exist in tree!");
   else if((*n)->data == i)
      printf("\nValue found!");
   else if(i > (*n)->data)
      searchNode(i, &((*n)->rightChildNode));
   else
      searchNode(i, &((*n)->leftChildNode));
}

int main()
{
   int ch, num, num1;
   clrscr();
   do {
      printf("\nSelect a choice from the menu below.");
      printf("\n1. Insert a node.");
      printf("\n2. Search for a node.");
      printf("\n3.exit\n");
      printf("\nChoice: ");
```

```
        scanf("%d", &ch);
        switch(ch) {
        case 1:
            printf("\nEnter an element: ");
            scanf("%d", &num);
            insertNode(num, &rootNode);
            break;
        case 2:
            printf("\nEnter the element to be searched for: ");
            scanf("%d", &num);
            searchNode(num, &rootNode);
            break;
        case 3:


           exit(0);
           default:
           printf("invalid choice");
        }
         } while(num!=3);
    getch();
    return 0;
}
```

Output

```
Select a choice from the menu below.
1. Insert a node.
2. Search for a node.
3.exit

Choice: 1

Enter an element: 7

Select a choice from the menu below.
1. Insert a node.
2. Search for a node.
3.exit

Choice: 1

Enter an element: 5

Select a choice from the menu below.
1. Insert a node.
2. Search for a node.
3.exit

Choice: 1_
```

```
Select a choice from the menu below.
1. Insert a node.
2. Search for a node.
3.exit

Choice: 1

Enter an element: 8

Select a choice from the menu below.
1. Insert a node.
2. Search for a node.
3.exit

Choice: 2

Enter the element to be searched for: 5

Value found!
Select a choice from the menu below.
1. Insert a node.
2. Search for a node.
3.exit

Choice: 3_
```