DATA STRUCTURES LAB USING C

(Course Code: MCACP207)

Submitted in partial fulfilment of the requirement for the award of the degree of

MASTER OF COMPUTER APPLICATION

SEMESTER II [2022-2024]

Submitted by

SEBIN THOMAS

Register No : 223242211039

Under the guidance of

Ms ANU JOSEPH

DEPARTMENT OF COMPUTER APPLICATION,

KRISTU JYOTI COLLEGE OF MANAGEMENT AND TECHNOLOGY,

CHANGANASSERY

September-2023

KRISTU JYOTI COLLEGE OF MANAGEMENT AND TECHNOLOGY

CHANGANASSERY

DEPARTMENT OF COMPUTER APPLICATION

CERTIFICATE

Certify that the practical laboratory record of DATA STRUCTURES USING C is a bona-fide report of the practical works done by SEBIN THOMAS(Reg No.223242211039) under our guidance and supervision is submitted in partial fulfilment of the Master of Computer Applications, awarded by Mahatma Gandhi University, Kerala.

Ms. Anu Joseph……..……..……

(Teacher-in-Charge)

Mr. Roji Thomas…….....

(H.O.D)

Rev.Fr. Joshy Cheeramkuzhy CMI……

(Principal)

Submitted for practical examination and viva held on…….

Department Seal

Examiner(s)

1.

2.

DECLARATION

I hereby declare that the project work entitled "DATA STRUCTURES USING C" submitted to Mahatma Gandhi University in partial fulfillment of requirement for the award of post-graduation of Master of Computer Application from Kristu Jyoti College of management and technology, Changanacherry is a record of bona-fide work done under the guidance of Ms Anu Joseph, Department of Computer Application. This project work has not been submitted in partial or fulfilment of any other post-graduation or similar of this University or any other university.

SEBIN THOMAS

Reg no: 223242211039

ACKNOWLEDGEMENT

I would like to express my thankfulness towards a number of people who have been instrument in making of this project. First of all, we thank the college management who has been with us with most helpful facilities throughout the completion of the project. I am greatly indebted to Rev. Fr. Joshy Cheeramkuzhy CMI, Principal of Kristu Jyoti College who whole heartedly give us the permission and whose advice was a real encouragement. I record my sincere thanks and gratitude to Mr. Roji Thomas, HOD, Computer Application, for the valuable suggestions provided. I am deeply indebted to Asst. Prof. Ms Anu Joseph Department of Computer Application, for the valuable discussion and helpful counselling. Last, not the least I wish to express my gratitude and heartfelt thanks to our friends, family and whom with their valuable advice, encouragement and support for the successful completion of the project. Above all thanking the GOD the almighty.

SEBIN THOMAS

CONTENT

16	Quick sort	**64**	**5/6/23**
17	Heap sort	**66**	**5/16/23**
18	Selection sort	**69**	**5/6/23**

BASIC PROGRAMS

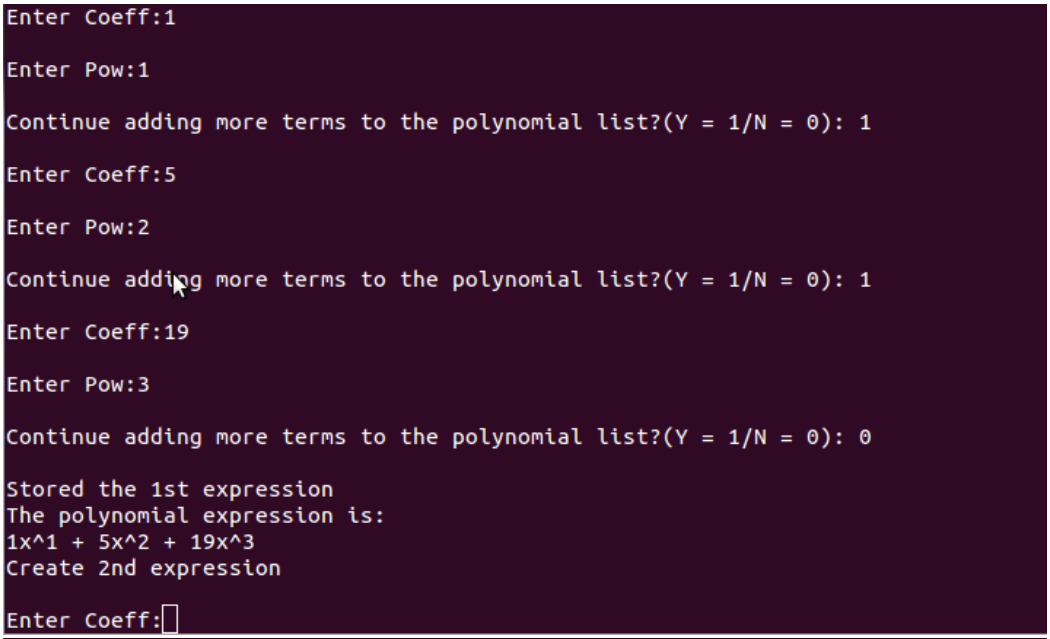
1.Write a program for polynomial addition

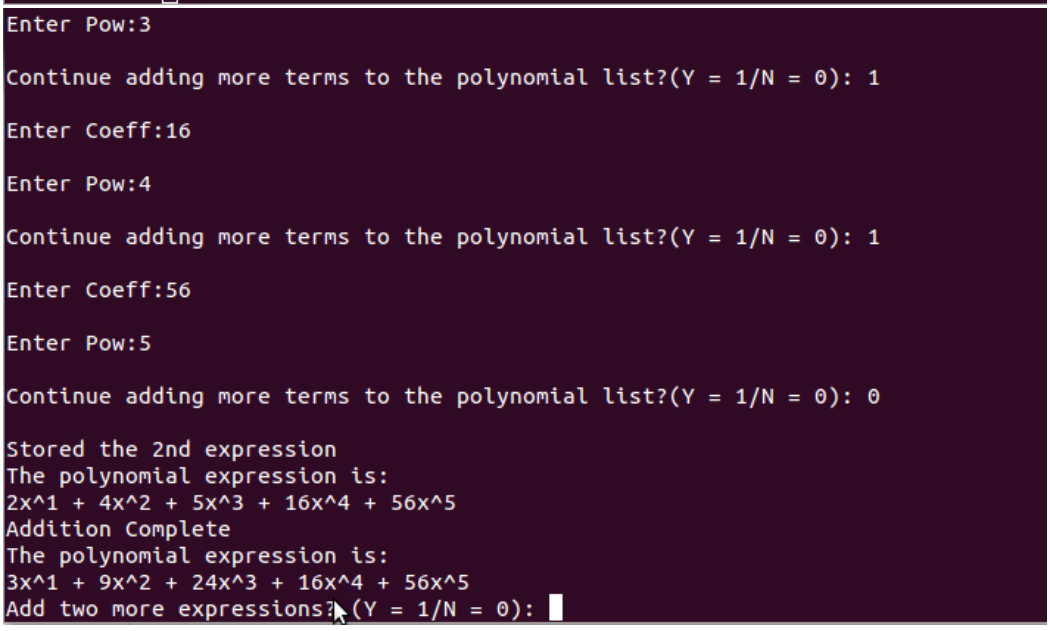
```
#include <stdio.h>
#include<stdlib.h>
struct Term
{
   int coeff;
   int exp;
};
struct Poly
{
   int n;
   struct Term *terms;
};
void create (struct Poly *p)
{
   int i;
   printf ("Enter Number of terms: ");
   scanf ("%d", &p->n);
   p->terms = (struct Term *) malloc (p->n * sizeof (struct Term));
   printf ("Enter terms:\n");
   for (i = 0; i < p->n; i++)
      scanf ("%d%d", &p->terms[i].coeff, &p->terms[i].exp);
   printf ("\n");
}
void display (struct Poly p)
{
   int i;
```

```
    for (i = 0; i < p.n; i++)
    {
        printf ("%dx%d", p.terms[i].coeff, p.terms[i].exp);
        if (i + 1 < p.n)
         printf (" + ");
    }
    printf ("\n");
}
struct Poly *add (struct Poly *p1, struct Poly *p2)
{
    int i, j, k;
    struct Poly *sum;
    sum = (struct Poly *) malloc (sizeof (struct Poly));
    sum->terms = (struct Term *) malloc ((p1->n + p2->n) * sizeof (struct Term));
    i = j = k = 0;
    while (i < p1->n && j < p2->n)
    {
        if (p1->terms[i].exp > p2->terms[j].exp)
         sum->terms[k++] = p1->terms[i++];
        else if (p1->terms[i].exp < p2->terms[j].exp)
         sum->terms[k++] = p2->terms[j++];
        else
     {
         sum->terms[k].exp = p1->terms[i].exp;
         sum->terms[k++].coeff = p1->terms[i++].coeff + p2->terms[j++].coeff;
     }
    }
    for (; i < p1->n; i++)
```

```
        sum->terms[k++] = p1->terms[i];
    for (; j < p2->n; j++)
        sum->terms[k++] = p2->terms[j];
    sum->n = k;
    return sum;
}
int main()
{
    struct Poly p1, p2, *p3;
    printf ("Enter Polynomial 1:\n");
    create (&p1);
    printf ("Enter Polynomial 2:\n");
    create (&p2);
    p3 = add (&p1, &p2);
    printf ("\n");
    printf ("Polynomial 1 is: ");
    display (p1);
    printf ("\n");
    printf ("Polynomial 2 is: ");
    display (p2);
    printf ("\n");
    printf ("Polynomial 3 is: ");
    display (*p3);
    return 0;
    getch();
}
```

Output





2.Implementation of stack

```
#include<stdio.h>
int stack[100],choice,n,top,x,i;
void push(void);
```

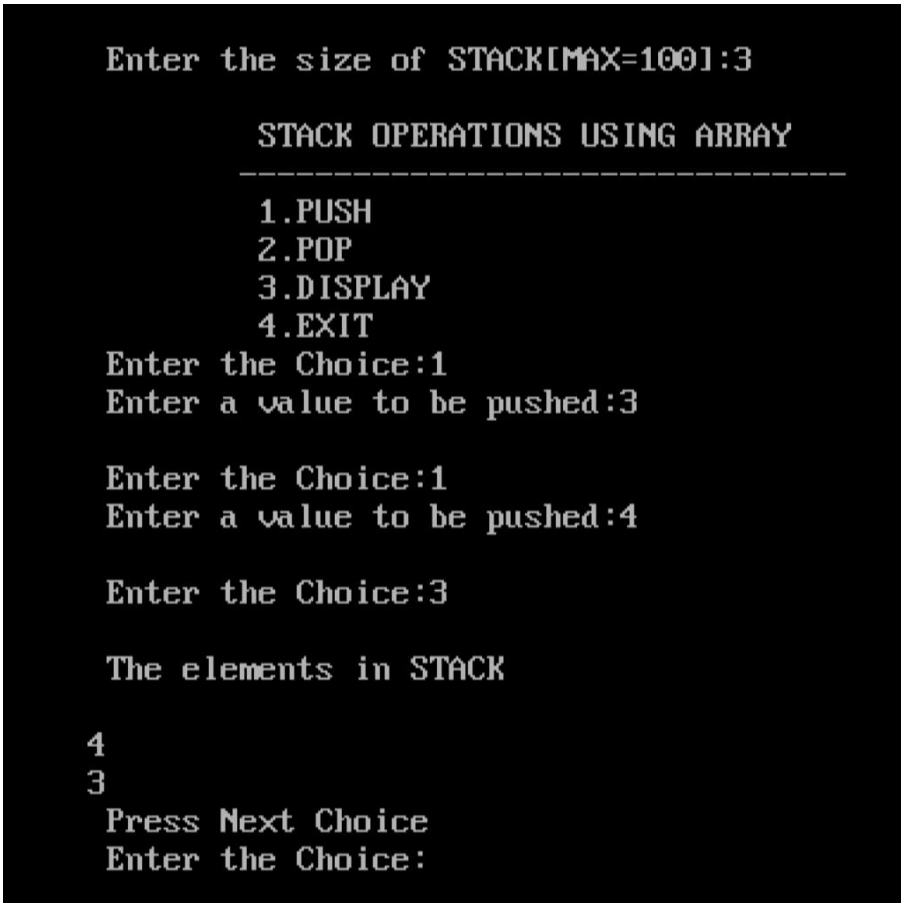
```
void pop(void);
void display(void);
int main()
{
   //clrscr();
   top=-1;
   printf("\n Enter the size of STACK[MAX=100]:");
   scanf("%d",&n);
   printf("\n\t STACK OPERATIONS USING ARRAY");
   printf("\n\t----------");
   printf("\n\t 1.PUSH\n\t 2.POP\n\t 3.DISPLAY\n\t 4.EXIT");
   do
   {
      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;
}
void push()
{
  if(top>=n-1)
  {
    printf("\n\tSTACK is over flow");

  }
  else
  {
```

```
        printf(" Enter a value to 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



3. Conversion of an infix expression to postfix expression

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

```
#include<stdlib.h>
#define MAX 20
char stack[MAX];
int top=1;
char pop();
void push(char item);
int prcd(char symbol)
{
 switch(symbol)
 {
 case '+':
 case '-': return 2;
 break;
 case '*':
 case '/': return 4;
 break;
 case '^':return 6;
 break;
 case '(':
 case ')':
 case '#':return 1;
 break;
 }
 return 0;
}
int(isoperator(char symbol))
{
 switch(symbol)
```

```
 {
 case '+':
 case '*':
 case '-':
 case '/':
 case '^':
 case '(':
 case ')':return 1;
 break;
 default:return 0;
 }
}
void convertip(char infix[],char postfix[])
{
int i,symbol,j=0;
stack[++top]='#';
for(i=0;i<strlen(infix);i++)
{
 symbol=infix[i];
 if(isoperator(symbol)==0)
 {
 postfix[j]=symbol;
 j++;
 }
 else
 {
 if(symbol=='(')
 push(symbol);
```

```
else if(symbol==')')
{
while(stack[top]!='(')
{
postfix[j]=pop();
j++;
}
pop();
}
else
{
if(prcd(symbol)>prcd(stack[top]))
push(symbol);
else
{
while(prcd(symbol)<=prcd(stack[top]))
{
postfix[j]=pop();
j++;
}
push(symbol);
}
}
}
}       }
While (stack[top]!='#')
{
postfix[j]=pop();
```

```
 j++;
 }
 postfix[j]='\0';
}
void main()
{
 char infix[20],postfix[20];
 printf("enter the valid infix string \n");
 gets(infix);
 convertip(infix,postfix);
 printf("the corresponding postfix string is:\n");
 puts(postfix);
 }
void push(char item)
{
 top++;
 stack[top]=item;
}
char pop()
{
 char a;
 a=stack[top];
 top--;
 return a;
}
```

Output

First Run:

Enter Infix expression: A+(B*C-(D/E^F)*G)*H

Postfix Expression: ABC*DEF^/G*-H*+

Second Run:

Enter Infix expression (3^2*5)/(3*2-3)+5

Postfix Expression: 32^5*32*3-/5+

4.Evaluating a postfix expression

```
#include<stdio.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;
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());
   getch();
   return 0;
}
```

Output

```
Enter the expression :: 25*+

The result of expression 25*+  =  10
```

5. Implementation of queue

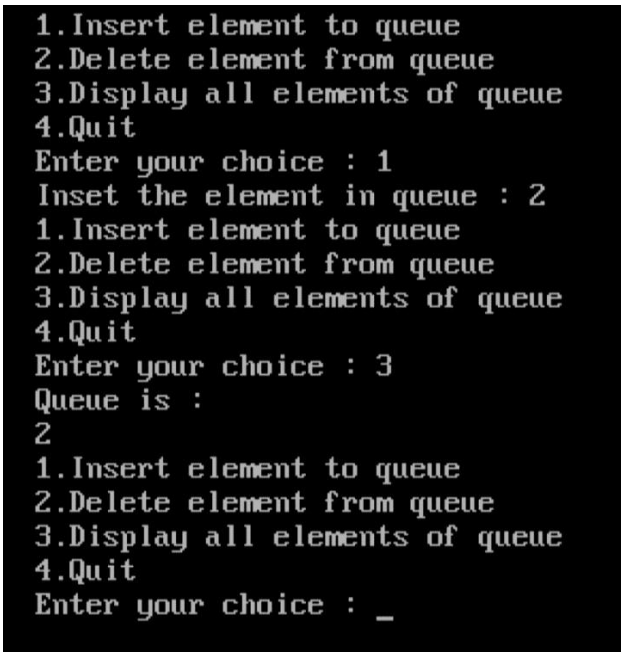
```
#include <stdio.h>

#define MAX 50

void    insert();
void   delete();
void display();
int queue_array[MAX];
int rear = - 1;
int front = - 1;
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:
            delete();
            break;
            case 3:
            display();
            break;
            case 4:
            exit(1);
            default:
            printf("Wrong choice \n");
        } /* End of switch */
    } /* End of while */
} /* End of main() */

void insert()
{
    int add_item;
    if (rear == MAX - 1)
    printf("Queue Overflow \n");
    else
    {
        if (front == - 1)
        /*If queue is initially empty */
        front = 0;
        printf("Inset the element in queue : ");
        scanf("%d", &add_item);
        rear = rear + 1;
        queue_array[rear] = add_item;
```

```
    }
} /* End of insert() */

void delete()
{
    if (front == - 1 || front > rear)
    {
        printf("Queue Underflow \n");
        return ;
    }
    else
    {
        printf("Element deleted from queue is : %d\n", queue_array[front]);
        front = front + 1;
    }
} /* End of delete() */

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_array[i]);
        printf("\n");
```

```
    }
} /* End of display() */n
```

Output



6.Implementation of Circular Queue

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#define MAXQ 100
int front=-1,rear=-1;
int items[MAXQ];
int Isempty();
int Isfull();
```

```
void Insert(int);
int Delete();
void Display();
void main()
{
int x;
char ch='1';
clrscr();
while(ch!='4')
{
printf("\n 1-INSERT");
printf("\n 2-DELETE");
printf("\n 3-DISPLAY");
printf("\n 4-QUIT");
printf("\n Enter your choice:");
fflush(stdin);
ch=getchar();
switch(ch)
{
case '1':
printf("\n Enter the nos of element to be inserted:");
scanf("%d",&x);
Insert(x);
break;
case '2':
x=Delete();
printf("\n Deleted element is %d\n:",x);
break;
```

```
case '3':
Display();
break;
case '4':
break;
default:
printf("\n Wrong choice!Try again:");
}
}
getch();
}
int Isempty()
{
if(front==-1)
return  1;
else
return 0;
}
int Isfull()
{
if(front==(rear+1)%MAXQ)
return 1;
else
return 0;
}
void Insert(int x)
{
if(Isfull())
```

```
{
printf("\n Queue full");
return;
}
if (front==-1)
{
front=0;
rear=0;
}
else
rear=(rear+1)%MAXQ;
items[rear]=x;
}
int Delete()
{
int x;
if(Isempty())
{
printf("\n Queue is empty");
exit(0);
}
x=items[front];
if (front==rear)
{
front=-1;
rear=-1;
}
else
```

```
front=(front+1)%MAXQ;
return x;
}
void Display()
{
int i,n;
if(Isempty())
{
printf("\n Queue is empty");
return;
}
printf("\n Elements in the Queue are :\n");
if(front<=rear)
{
for(i=front;i<=rear;i++)
printf("%d\n",items[i]);
}
else
{
for(i=front;i<=MAXQ-1;i++)
printf("%d\n",items[i]);
for(i=0;i<=rear;i++)
printf("%d\n",items[i]);
}
}
```

Output

```
 1-INSERT
 2-DELETE
 3-DISPLAY
 4-QUIT
 Enter your choice:1

 Enter the element to be inserted:4

 1-INSERT
 2-DELETE
 3-DISPLAY
 4-QUIT
 Enter your choice:1

 Enter the element to be inserted:3

 1-INSERT
 2-DELETE
 3-DISPLAY
 4-QUIT
 Enter your choice:_
```

```
 Elements in the Queue are :
4
3

 1-INSERT
 2-DELETE
 3-DISPLAY
 4-QUIT
 Enter your choice:
```

7.Implementation of linked list and performing insertions and deletions at both ends and also in between

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

```
#include<process.h>
struct node
{
int info;
struct node *next;
};
struct node *start=NULL;
void ins();
void  ins_at_beg();
void ins_at_mid();
void  ins_at_end();
void del();
void  del_at_beg();
void del_at_mid();
void  del_at_end();
void display();
int count();
void main()
{
int ch=0,i=0,cnt;
clrscr();
while(1)
{
 printf("**********menu**********");
 printf("\n1.insert");
 printf("\n2.delete");
 printf("\n3.display");
 printf("\n4.count");
```

```
 printf("\n5.exit");
 printf ("\nenter your choice : ");
 scanf("%d",&ch);
 switch(ch)
 {
 case 1:ins();
 break;
 case 2:del();
 break;
 case 3:display();
 break;
 case 4:cnt=count();
 printf("\n the no of nodes : %d\n",cnt);
 break;
 case 5:exit(1);
 }
 }
}
void ins()
{
 int j=0,ch1=0;
 printf("\nenter your choice");
 printf("\n1.insert at the beggning");
 printf("\n2.insert at the middle");
 printf("\n3.insert at the end");
 scanf ("%d",&ch1);
 switch(ch1)
 {
```

```
 case 1:ins_at_beg();
 break;
 case 2:ins_at_mid();
 break;
 case 3:ins_at_end();
 }
}
void ins_at_beg()
{
 int info;
 struct node *t=(struct node *)malloc(sizeof(struct node));
 printf("\nenter information to be inserted in the beggning");
 scanf("%d",&info);
 t->info=info;
 t->next=start;
 start=t;
}
void ins_at_mid()
{
 int inform,x,i;
 struct node *t=(struct node *)malloc(sizeof(struct node));
 struct node *p=start;
 printf("\nenter the location after which new node to be added");
 scanf("%d",&x);
 for(i=1;i<x;i++)
 p=p->next;
 printf("\nenter information of the new node");
 scanf("%d",&inform);
```

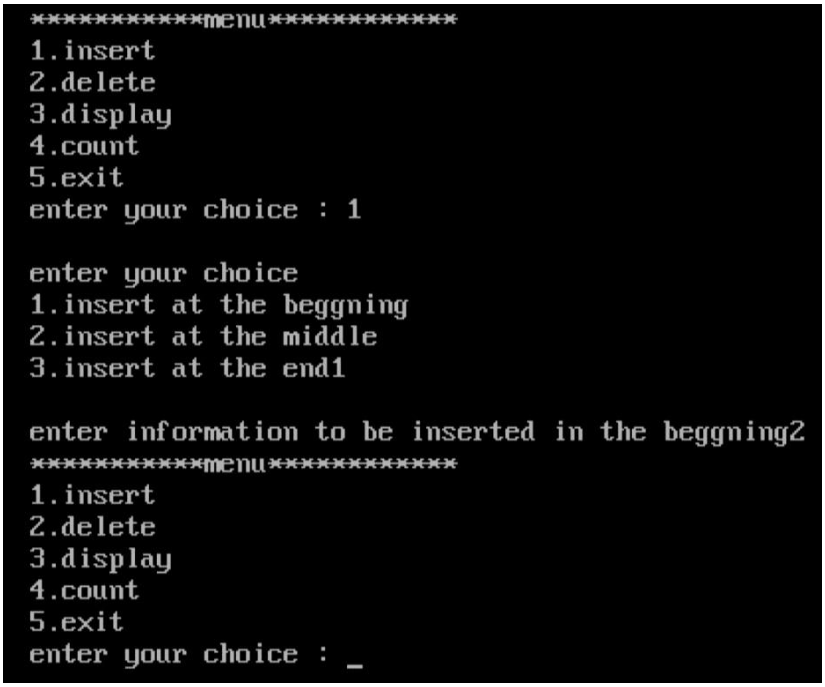
```
 t->info=inform;
 t->next=p->next;
 p->next=t;
}
void ins_at_end()
{
 int inform1;
 struct node *t=(struct node *)malloc(sizeof(struct node));
 struct node *p=start;
 printf("\nenter information to be added");
 scanf("%d",&inform1);
 t->info=inform1;
 while(p->next!=NULL)
 p=p->next;
 p->next=t;
 t->next=NULL;
}
void del()
{
int k=0,ch2=0;
 printf("\nenter your choice");
 printf("\n1.delete at the beggning");
 printf("\n2.delete at the middle");
 printf("\n3.delete at the end");
 scanf ("%d",&ch2);
 switch(ch2)
 {
 case 1:del_at_beg();
```

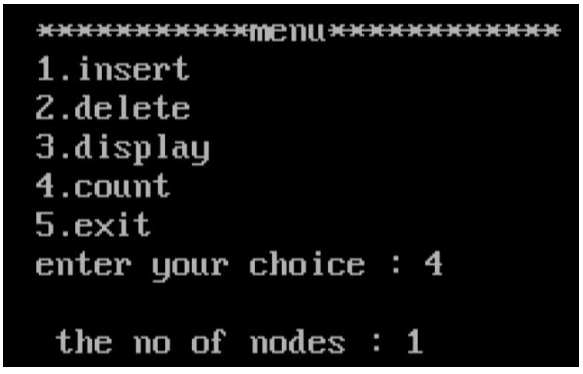
```
 break;
 case 2:del_at_mid();
 break;
 case 3:del_at_end();
 break;
 }
}
void del_at_beg()
{
 struct node *t=start;
 start=start->next;
 free(t);
}
void del_at_mid()
{
 int n;
 struct node *cur=start;
 struct node *pre=start;
 printf("\nenter information to be deleted");
 scanf("%d",&n);
 while(cur->info!=n)
 {
 pre=cur;
 cur=cur->next;
 }
 pre->next=cur->next;
 free(cur);
}
```

```
void del_at_end()
{
 struct node *cur=start;
 struct node *pre=start;
 while(cur->next!=NULL)
 {
 pre=cur;
 cur=cur->next;
 }
 pre->next=NULL;
 free(cur);
}
void display()
{
 struct node *p=start;
 printf("\n\n**********LINK LIST**********\n\n");
 while(p!=NULL)
 {
 printf("%d\n",p->info);
 p=p->next;
 }

}
int count()
{
 int c=0;
 struct node *q=start;
 while(q!=NULL)
```

```
{
q=q->next;

c=c+1;
}
 return c;
}
```

Output





8.Implementation of stack using Linked List

```
#include  <stdio.h>
#include <stdlib.h>
struct node {
 int info;
 struct node *ptr;
}*top,*top1,*temp;
int topelement();
void push(int data);
void pop();
void empty();
void display();
void destroy();
void stack_count();
void create();
int count = 0;
void main() {
int no, ch, e;
 printf("\n 1 - Push");
```

```
 printf("\n 2 - Pop");
 printf("\n 3 - Top");
 printf("\n 4 - Empty");
 printf("\n 5 - Exit");
 printf("\n 6 - Dipslay");
 printf("\n 7 - Stack Count");
 printf("\n 8 - Destroy stack");
 create();
 while (1)
{
 printf("\n Enter choice : ");
 scanf("%d", &ch);
 switch (ch)
{
 case 1:
 printf("Enter data : ");
 scanf("%d", &no);
 push(no);
 break;
 case 2:
 pop();
break;
case 3:
 if (top == NULL)
 printf("No elements in stack");
 else
{
 e = topelement();
```

```
 printf("\n Top element : %d", e);
}
 break;
 case 4:
 empty();
 break;
 case 5:
 exit(0);
 case 6:
 display();
 break;
 case 7:
 stack_count();
 break;
 case 8:
 destroy();
 break;
 default :
 printf(" Wrong choice, Please enter correct choice ");
 break;
}
}
}
/* Create empty stack */
void create() {
 top = NULL; }
/* Count stack elements */
void stack_count() {
```

```
 printf("\n No. of elements in stack : %d", count);
}
/* Push data into stack */
void push(int data) {
 if (top == NULL)
{
 top =(struct node *)malloc(1*sizeof(struct node));
 top->ptr = NULL;
 top->info = data;
}
 else
{
 temp =(struct node *)malloc(1*sizeof(struct node));
 temp->ptr = top;
 temp->info = data;
 top = temp;
}
count++; }
/* Display stack elements */
void display() {
 top1 = top;
 if (top1 == NULL)
{
 printf("Stack is empty");
 return;
}
 while (top1 != NULL)
{
```

```
 printf("%d ", top1->info);
 top1 = top1->ptr;
}
}
/* Pop Operation on stack */
void pop() {
 top1 = top;
 if (top1 == NULL)
{
 printf("\n Error : Trying to pop from empty stack");
 return;
}
 else
 top1 = top1->ptr;
 printf("\n Popped value : %d", top->info);
 free(top);
 top = top1;
 count--;
}
/* Return top element */
int topelement() {
return(top->info);
}
/* Check if stack is empty or not */
void empty() {
 if (top == NULL)
 printf("\n Stack is empty");
 else
```

```
 printf("\n Stack is not empty with %d elements", count);
}
/* Destroy entire stack */
void destroy() {
 top1 = top;
while (top1 != NULL)
{
 top1 = top->ptr;
 free(top);
 top = top1;
 top1 = top1->ptr;
}
 free(top1);
 top = NULL;
 printf("\n All stack elements destroyed");
 count = 0;
}
```

Output

9. Implementation of queue using Linked List

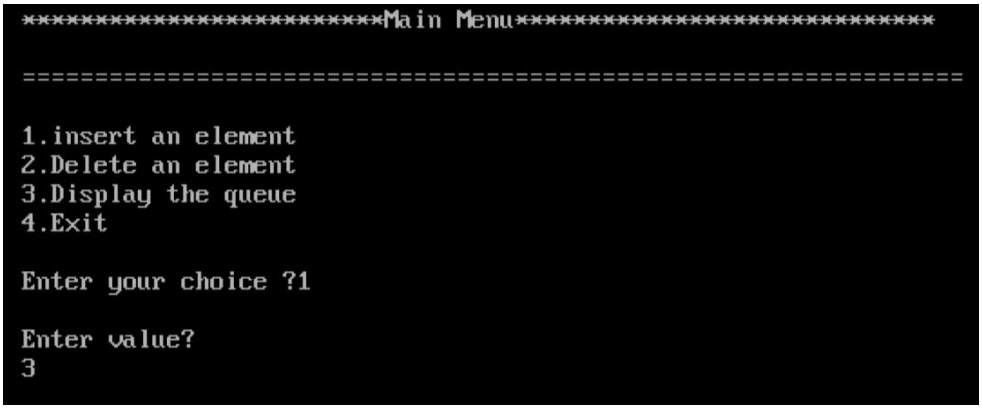
```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
struct node
{
int data;
struct node *next;
};
struct node *front;
struct node *rear;
void insert();
void delete();
void display();
void main ()
{
int choice;
while(choice != 4)
{
printf("\n*************************Main Menu*****************************\n");
printf("\n==========================================================
===\n");
printf("\n1.insert an element\n2.Delete an element\n3.Display the queue\n4.Exit\n");
printf("\nEnter your choice ?");
scanf("%d",&choice);
switch(choice)
{
```

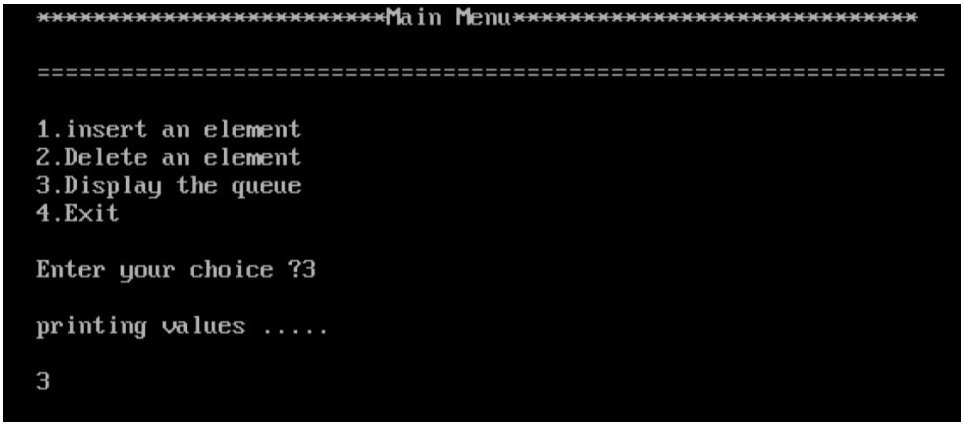
```
case 1:
insert();
break;
case 2:
delete();
break;
case 3:
display();
break;
case 4:
exit(0);
break;
default:
printf("\nEnter valid choice??");
}
}
}

void insert()
{
struct node *ptr;
int item;
ptr = (struct node *) malloc (sizeof(struct node));
if(ptr == NULL)
{
printf("\nOVERFLOW\n");
return;
}
```

```
else
{
printf("\nEnter value?\n");
scanf("%d",&item);
ptr -> data = item;
if(front == NULL)
{
front = ptr;
rear = ptr;
front -> next = NULL;
rear -> next = NULL;
}
else
{
rear -> next = ptr;
rear = ptr;
rear->next = NULL;
}
}
}
void delete ()
{
struct node *ptr;
if(front == NULL)
{
printf("\nUNDERFLOW\n");
return;
}
```

```
else
{
ptr = front;
front = front -> next;
free(ptr);
}
}

void display()
{
struct node *ptr;
ptr =  front;
if(front == NULL)
{
printf("\nEmpty queue\n");
}
else
{ printf("\nprinting values .....\n");
while(ptr != NULL)
{
printf("\n%d\n",ptr -> data);
ptr = ptr -> next;
}
}
}
```

Output





10. Implementation of a doubly linked list.

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
 struct node *prev;
 int data;
 struct node *next;
};
struct node *start,*nt;
void insertbeg(void)
```

```
{
 int a;
 struct node *nn,*temp;
 nn=(struct node *)malloc(sizeof(struct node));
 printf("enter data:");
 scanf("%d",&nn->data);
 a=nn->data;
 if(start==NULL) /*checking if List is empty*/
 {
 nn->prev=nn->next=NULL;
 start=nn;
 }
 else
{
 nn->next=start;
 nn->prev=NULL;
 start->prev=nn;
 start=nn;
 }
 printf("%d succ inserted \n",a);
}
void insertend(void)
{
 int b;
 struct node *nn,*lp;
 nn=(struct node *)malloc(sizeof(struct node));
 printf("enter data:");
 scanf("%d",&nn->data);
```

```
 b=nn->data;
 if(start==NULL)
 {
 nn->prev=nn->next=NULL;
 start=nn;
 }
 else
{
 lp=start;
 while(lp->next!=NULL)
 {
 lp=lp->next;
 }
 nn->prev=lp;
 lp->next=nn;
 nn->next=NULL;
}
printf("%d succ inserted\n",b);
}
void insertmid(void)
{
 struct node *nn,*temp,*ptemp;
 int x,c;
 if(start==NULL)
 {
 printf("dll is empty\n");
 return;
 }
```

```
printf("enter data before which nn is to be inserted\n");
scanf("%d",&x);
if(x==start->data)
{
insertbeg();
}
ptemp=start;
temp=start->next;
while(temp->next!=NULL&&temp->data!=x)
{
ptemp=temp;
temp=temp->next;
}
if(temp==NULL)
{
printf("%d does not exit\n",x);
}
else
{
nn=(struct node *)malloc(sizeof(struct node));
printf("enter data");
scanf("%d",&nn->data);
c=nn->data;
nn->data;
nn->prev=ptemp;
nn->next=temp;
ptemp->next=nn;
temp->prev=nn;
```

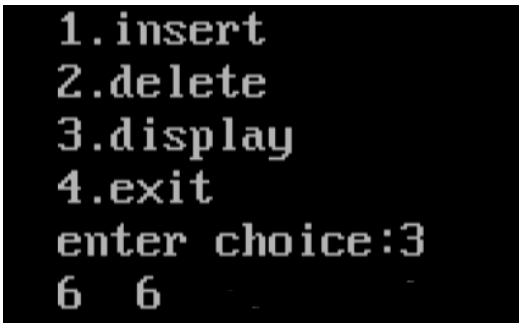
```
printf("%d succ inserted \n",c);
}
}
void deletion() {
struct node *pt,*t;
int x;
t=pt=start;
if(start==NULL)
{
printf("dll is empty\n");
}
printf("enter data to be deleted:");
scanf("%d",&x);
if(x==start->data)
{
t=start;
t=t->next;
free(start);
start=t;
start=pt;
}
else {
while(t->next!=NULL&&t->data!=x)
{
pt=t;
t=t->next;
}
if(t->next==NULL&&t->data==x)
```

```
{
 free(t);
 pt->next=NULL;
}
else {
 if(t->next==NULL&&t->data!=x)
 printf("data not found");
 else
{
 pt->next=t->next;
free(t);
 }
}
printf("%d is succ deleted\n",x);
}
}
void display()
{
 struct node *temp;
 if(start==NULL)
 printf("stack is empty ");
 temp=start;
 while(temp->next!=NULL)
 {
 printf("%d  ",temp->data);
 temp=temp->next;
 }
 printf("%d  ",temp->data);
```

```
}
int main()
{
 int c,a;
 start=NULL;
 do
 {
 printf("1.insert\n2.delete\n3.display\n4.exit\nenter choice:");
 scanf("%d",&c);
 switch(c)
 {
 case 1:printf("1.insertbeg\n2.insertend\n3.insertmid\nenter choice:");
 scanf("%d",&a);
 switch(a)
 {
 case 1:insertbeg();
 break;
 case 2:insertend();
 break;
 case 3:insertmid();
 break;
 }
 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;
}
```

Output



11. Creation of a binary tree and binary search tree and performing the traversals.

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

typedef struct BST {
  int data;
  struct BST *lchild, *rchild;
} node;

void insert(node *, node *);
void inorder(node *);
void preorder(node *);
void postorder(node *);
node *search(node *, int, node **);

void main() {
  int choice;
  char ans = 'N';
  int key;
  node *new_node, *root, *tmp, *parent;
  node *get_node();
  root = NULL;
  clrscr();

  printf("\nProgram For Binary Search Tree ");
```

```
do {
  printf("\n1.Create");
  printf("\n2.Search");
  printf("\n3.Recursive Traversals");
  printf("\n4.Exit");
  printf("\nEnter your choice :");
  scanf("%d", &choice);

  switch (choice) {
  case 1:
    do {
      new_node = get_node();
      printf("\nEnter The Element ");
      scanf("%d", &new_node->data);

      if (root == NULL) /* Tree is not Created */
        root = new_node;
      else
        insert(root, new_node);

      printf("\nWant To enter More Elements?(y/n)");
      ans = getch();
    } while (ans == 'y');
    break;

  case 2:
    printf("\nEnter Element to be searched :");
    scanf("%d", &key);
```

```
        tmp = search(root, key, &parent);
        printf("\nParent of node %d is %d", tmp->data, parent->data);
        break;

      case 3:
        if (root == NULL)
          printf("Tree Is Not Created");
        else {
          printf("\nThe Inorder display : ");
          inorder(root);
          printf("\nThe Preorder display : ");
          preorder(root);
          printf("\nThe Postorder display : ");
          postorder(root);
        }
        break;
      }
    } while (choice != 4);
}
/*
Get new Node
*/
node *get_node() {
  node *temp;
  temp = (node *) malloc(sizeof(node));
  temp->lchild = NULL;
  temp->rchild = NULL;
```

```
  return temp;
}
/*
This function is for creating a binary search tree
*/
void insert(node *root, node *new_node) {
  if (new_node->data < root->data) {
    if (root->lchild == NULL)
      root->lchild = new_node;
    else
      insert(root->lchild, new_node);
  }

  if (new_node->data > root->data) {
    if (root->rchild == NULL)
      root->rchild = new_node;
    else
      insert(root->rchild, new_node);
  }
}
/*
This function is for searching the node from
binary Search Tree
*/
node *search(node *root, int key, node **parent) {
  node *temp;
  temp = root;
  while (temp != NULL) {
```

```
      if (temp->data == key) {
        printf("\nThe %d Element is Present", temp->data);
        return temp;
      }
      *parent = temp;

      if (temp->data > key)
        temp = temp->lchild;
      else
        temp = temp->rchild;
    }
    return NULL;
}
/*
This function displays the tree in inorder fashion
*/
void inorder(node *temp) {
    if (temp != NULL) {
      inorder(temp->lchild);
      printf("%d  ", temp->data);
      inorder(temp->rchild);
    }
}
/*
This function displays the tree in preorder fashion
*/
void preorder(node *temp) {
    if (temp != NULL) {
```

```
    printf("%d  ", temp->data);
    preorder(temp->lchild);
    preorder(temp->rchild);
  }
}

/*
This function displays the tree in postorder fashion
*/
void postorder(node *temp) {
  if (temp != NULL) {
    postorder(temp->lchild);
    postorder(temp->rchild);
    printf("%d  ", temp->data);
  }
}
```

Output

```
Program For Binary Search Tree
1.Create
2.Search
3.Recursive Traversals
4.Exit
Enter your choice :1

Enter The Element 2

Want To enter More Elements?(y/n)
Enter The Element 5

Want To enter More Elements?(y/n)
Enter The Element 9

Want To enter More Elements?(y/n)
Enter The Element
4
```

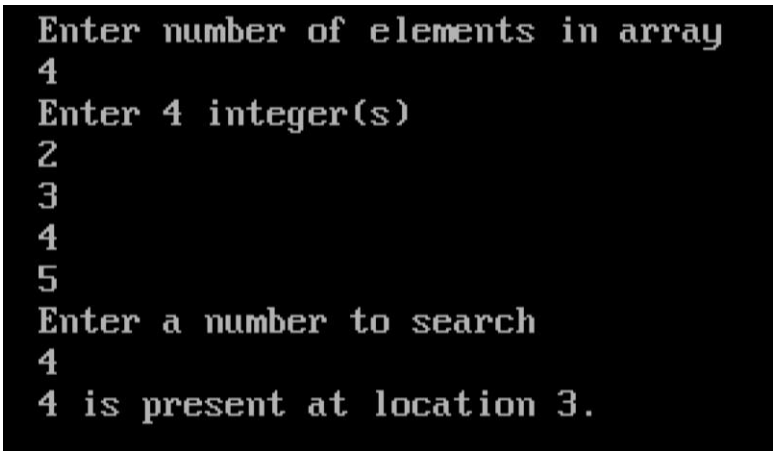
```
1.Create
2.Search
3.Recursive Traversals
4.Exit
Enter your choice : 2

Enter Element to be searched :5

The 5 Element is Present
Parent of node 5 is 2
```

12. Linear search.

```
#include <stdio.h>
#include<conio.h>
void main()
{
  int array[100], search, i, n;

  printf("Enter number of elements in array\n");
  scanf("%d", &n);

  printf("Enter %d integer(s)\n", n);

  for (i = 0; i < n; i++)
    scanf("%d", &array[i]);

  printf("Enter a number to search\n");
  scanf("%d", &search);

  for (i = 0; i < n; i++)
  {
```

```
    if (array[i] == search)    /* If required element is found */
    {
      printf("%d is present at location %d.\n", search, i+1);
      break;
    }
  }
  if (i == n)
    printf("%d isn't present in the array.\n", search);

  getch();
}
```

Output

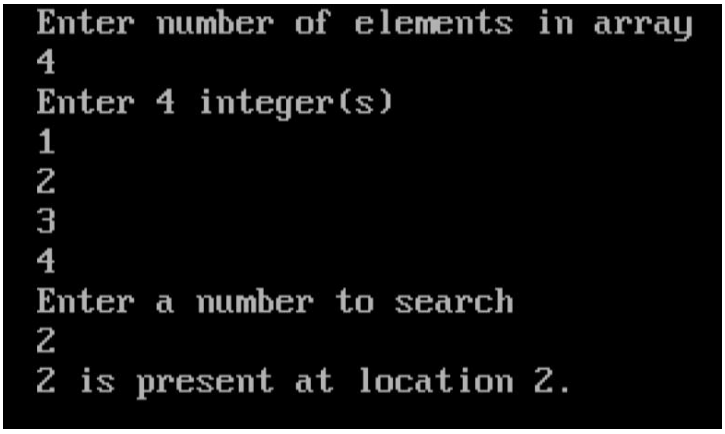


13. Binary search.

```
#include<stdio.h>
#include<conio.h>
void main()
```

```
{
int i, low, high, mid, n, key, array[100];
printf("Enter number of elements\n");
scanf("%d",&n);
printf("Enter %d integers\n", n);
for(i = 0; i < n; i++)
scanf("%d",&array[i]);
printf("Enter value to find\n");
scanf("%d", &key);
low = 0;
high = n - 1;
mid = (low+high)/2;
while (low <= high) {
if(array[mid] < key)
low = mid + 1;
else if (array[mid] == key) {
printf("%d found at location %d\n", key, mid+1);
break;
}
else
high = mid - 1;
mid = (low + high)/2;
}
if(low > high)
printf("Not found! %d isn't present in the list.n", key);
getch();
}
```

Output

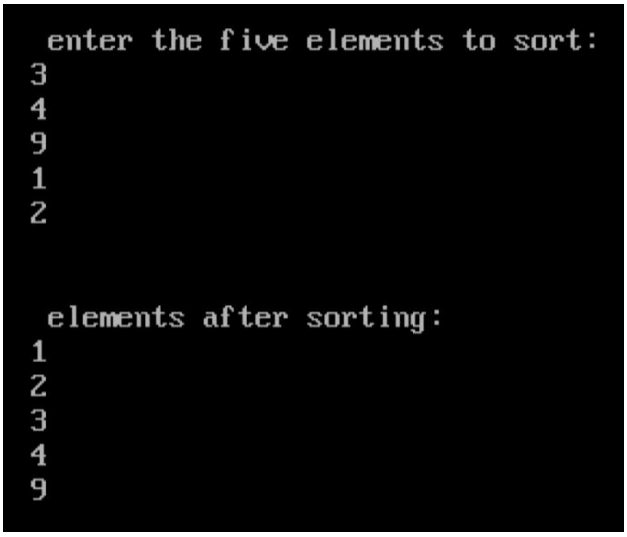


14. Insertion sort.

```
#include<stdio.h>
#include<conio.h>
void insertsort(int[]);
void main() {
 int num[5],count;
 printf("\n enter the five elements to sort:\n");
 for(count=0;count<5;count++)
 scanf("%d",&num[count]);
 insertsort(num); /*function call for insertion sort*/
 printf("\n\n elements after sorting:\n");
 for(count=0;count<5;count++)
 printf("%d\n",num[count]);
 getch();
}
void insertsort(int num[]) { /* function definition for insertion sort*/
```

```
int i,j,k;
for(j=1;j<5;j++) {
k=num[j];
for(i=j-1;i>=0&&k<num[i];i--)
num[i+1]=num[i];
num[i+1]=k; }
}
```

Output



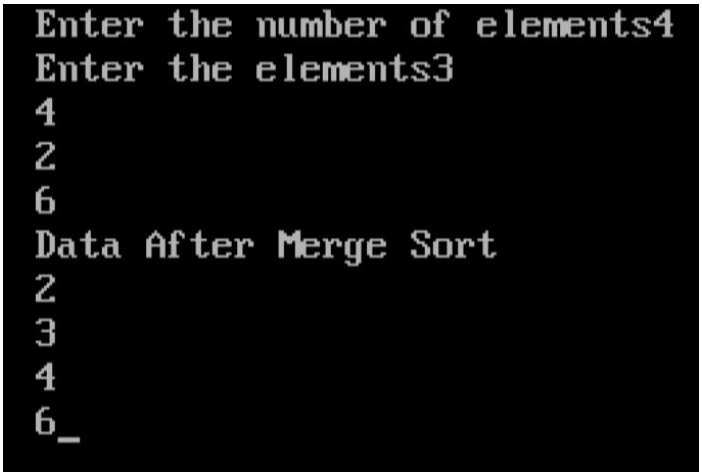
15. Merge sort.

```
#include<stdio.h>
#include<conio.h>
void mergesort(int a[],int,int);
void merge(int [],int,int,int);
void main()
{
```

```
 int a[20],i,n;
 clrscr();
 printf("Enter the number of elements");
 scanf("%d",&n);
 printf("Enter the elements");
 for(i=0;i<n;i++)
 {
 scanf("%d",&a[i]);
 }
 mergesort(a,0,n-1);
 printf("Data After Merge Sort");
 for(i=0;i<n;i++)
 printf("\n%d",a[i]);
 getch();
}
void mergesort(int a[],int lb,int ub)
{
 int mid;
 if(lb<ub)
 {
 mid=(lb+ub)/2;
 mergesort(a,lb,mid);
 mergesort(a,mid+1,ub);
 merge(a,lb,mid+1,ub);
 }
}
void merge(int a[],int lb,int mid,int ub)
{
```

```
int k,p1,p2,p3,b[20];
p1=lb;
p3=lb;
p2=mid;
while((p1<mid)&&(p2<=ub))
{
if(a[p1]<=a[p2])
b[p3++]=a[p1++];
else
b[p3++]=a[p2++];
}
while(p1<mid)
{
b[p3++]=a[p1++];
}
while(p2<=ub)
{
b[p3++]=a[p2++];
}
for(k=lb;k<p3;k++)
{
a[k]=b[k];
}
}
```

Output



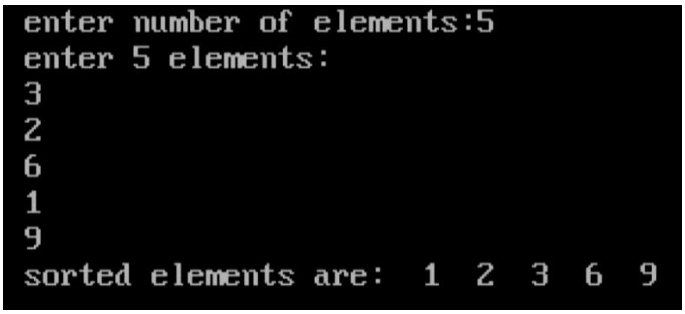
16. Quick sort.

```
#include<stdio.h>
#include<conio.h>
void quicksort(int x[10],int,int);
void main()
{
 int x[10],i,n;
 printf("enter number of elements:");
 scanf("%d",&n);
 printf("enter %d elements:\n",n);
 for(i=0;i<n;i++)
 scanf("%d",&x[i]);
 quicksort(x,0,n-1);/*function call*/
 printf("sorted elements are:");
 for(i=0;i<n;i++)
```

```
 printf("%3d",x[i]);
 getch();
}
/*called function*/
void quicksort(int x[10],int first,int last)
{
 int pivot,i,j,t;
 if(first<last)
 {
 pivot=first;
 i=first;
 j=last;
 while(i<j)
 {
 while(x[i]<=x[pivot]&&i<last)
 i++;
 while(x[j]>x[pivot])
 j--;
 if(i<j)
{
 t=x[i];
 x[i]=x[j];
 x[j]=t;
}
}
 t=x[pivot];
 x[pivot]=x[j];
 x[j]=t;
```

```
 quicksort(x,first,j
-1);
 quicksort(x,j+1,last);
}
}
```

Output



17. Heap sort.

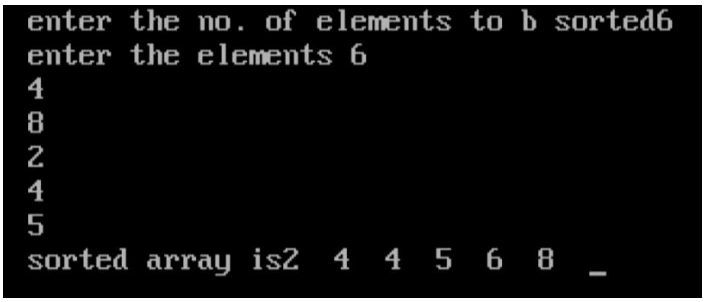
```
#include<stdio.h>
#include<conio.h>
int p(int);
int left(int);
int right(int);
void heapify(int[],int,int);
void buildheap(int[],int);
void heapsort(int[],int);
void main()

{
 int x[20],n,i;
```

```
printf("enter the no. of elements to b sorted");
scanf("%d",&n);
printf("enter the elements ");
for(i=0;i<n;i++)
scanf("%d",&x[i]);
heapsort(x,n);
printf("sorted array is");
for(i=0;i<n;i++)
printf("%d ",x[i]);
getch();
}
int p(int i)

{
return i/2;

}
int left(int i) {
return 2*i+1;

}
int right(int i)

{
return 2*i+2;

}
void heapify(int a[],int i,int n)
```

```
{
 int l,r,large,t;
 l=left(i);
 r=right(i);
 if((l<=n
-1)&&(a[l]>a[i]))
 large=l;
 else
 large=i;
 if((r<=n
-1)&&(a[r]>a[large]))
 large=r;
 if(large!=i)

{
 t=a[i];
 a[i]=a[large];
 a[large]=t;
 heapify(a,large,n);

}
}
void buildheap(int a[],int n) {
 int i;
 for(i=(n
-1)/2;i>=0;i--
)
```

```
 heapify(a,i,n); }
void heapsort(int a[],int n) {
 int i,m,t;
 buildheap(a,n);
 m=n ;
 for(i=n-1;i>=1;i--)
{
t=a[0];
 a[0]=a[i];
 a[i]=t;
 m=m-1;
 heapify(a,0,m);
 }
}
```

Output



18. Selection Sort

```
#include<stdio.h>
#include<conio.h>
void selection_sort(int a[],int n);
void main()
```

```
{
int a[100],i,n;
clrscr();
printf(“How many elements : ”);
scanf(“%d”,&n);
for(i=0;i<n;i++)
{
scanf(“%d”,&a[i]);
}
selection_sort(a,n);
printf()(“\nSorted array : ”);
for(i=0;i<n;i++)
{
printf(“\n%d”,a[i]);
}
getch();
}
void selection_sort(int a[],int n)
{
int min,loc,temp,i,j;
min=a[0];
for(i=0;i<n;i++)
{
min=a[i];
loc=i;
for(j=j+1;j<=n;j++)
{
if(a[j]<min)
```

```
{
min=a[j];
loc=j;
}
}
if(loc!=i)
{
temp=a[i];
a[i]=a[loc];
a[loc]=temp;
}
}
}
```

Output