1.WRITE A C PROGRAM TO ILLUSTRATE THE USE OF POINTERS IN ARITHMETIC OPERATIONS.

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

void main()
{
      int a[5] = {10, 20, 30, 40, 50} ;
      int *p1, *p2, i;

      clrscr();

      p1 = a ;

      printf("\n a=%u, p1=%u", a, p1);
      printf("\n First element using array name a[0]: %d", a[0]);
      printf("\n First element using pointer to array: %d", *p1);

      p1 = p1 + 1 ;

      printf("\n After incrementing pointer by 1: p1=%u, *p1=%d",p1,
*p1);

      p1 = p1 - 1 ;

      printf("\n After decrementing pointer by 1: p1=%u, *p1=%d",p1,
*p1);

      p1 = p1 + 4 ;

      printf("\n After incrementing pointer by 4: p1=%u, *p1=%d",p1,
*p1);

      p2 = a ;
```

```
     printf("\n After subtracting p2(%u) from p1(%u) p1-p2 =
%d",p2, p1, p1-p2);


     getch();
}
```

OUTPUT:

```
a=6487536, p1=6487536
 First element using array name a[0]: 10
 First element using pointer to array: 10
 After incrementing pointer by 1: p1=6487540, *p1=20
 After decrementing pointer by 1: p1=6487536, *p1=10
 After incrementing pointer by 4: p1=6487552, *p1=50
 After subtracting p2(6487536) from p1(6487552) p1-p2 = 4
```

2 . WRITE A C PROGRAM TO SWAP TWO NUMBERS USING CALL BY VALUE AND CALL BY REFERENCE PARAMETER PASSING TECHNIQUES.

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



int main()
{
      int a, b ;
      clrscr();

      printf("\nEnter two integers: ");
      scanf("%d%d", &a, &b);

      printf("\nValues before swap: a=%d, b=%d", a, b);

      a = a + b ;
      b = a - b ;
      a = a - b ;

      printf("\nValues after swap: a=%d, b=%d", a, b);

      getch();
      return 0;
}
```

OUTPUT:

```
Enter two integers: 10 20

Values before swap: a=10, b=20
Values after swap: a=20, b=10
```

3. WRITE A C PROGRAM TO FIND THE SMALLEST ELEMENT IN AN ARRAY OF 10 ELEMENTS USING POINTERS.

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

void main()
{
      int a[10], i, *ptr, *small;
      clrscr() ;
      printf("\n Enter 10 integers:\n") ;
      ptr = a ;
      for(i=0; i<10; i++)
      {
            scanf("%d", ptr) ;
            ptr++;
      }
      small = a ;
      ptr = small + 1 ;
      for(i=1; i<10; i++)
      {
            if(*ptr < *small)
                  small = ptr ;
            ptr++ ;
      }
      printf("\n The smallest elment in the array = %d ", *small);
      getch() ;
}
```

OUTPUT:

```
Enter 10 integers:
80 90 45 30 77 89 13 66 82 100

 The smallest elment in the array = 13
```

4 . WRITE A C PROGRAM TO CREATE A DYNAMIC ARRAY OF INTEGERS USING POINTERS.

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

      int *a,n,i;
      clrscr();

      printf("\n enter the size of the array\n");
      scanf("\n%d", &n);

      a=(int*)malloc(n* sizeof (int));

      printf("\n enter the array element:\n");

      for(i=0;i<n;i++)
      scanf("%d",a+i);

      printf("\n array contents:\n");
      for(i=0;i<n;i++)
      printf("\n%d", *(a+i));

      free(a);
      getch();
}
```

OUTPUT:

```
enter the size of the array
5

 enter the array element:
12 65 10 76 99

 array contents:

12
65
10
76
99
```

5 . WRITE A C PROGRAM TO ILLUSTRATE THE USE OF FUNCTION POINTER.

```
#include<stdio.h>

void showme()
{
      printf("\n showme() function is called! \n");
}

void odd()
{
      printf("\n The number is odd \n");
}
void even()
{
      printf("\n The number is even \n");
}

int main()
{
      int *p; //int pointer declaration
      int x;
      void (*ptr)() ; //declaration of pointer to function return
nothing
      //showme();

      ptr = showme ;
      (*ptr)() ;

      printf("\n Enter X: ");
      scanf("%d", &x);

      if(x%2 == 0)
```

```
            ptr = even ;
      else
            ptr = odd ;

      (*ptr)();

      return 0;
}
```

OUTPUT:

```
showme() function is called!

 Enter X: 10

 The number is even
```

6 . WRITE A C PROGRAM TO COUNT THE NUMBER OF CHARACTERS IN A GIVEN FILE.

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

void main()
{
      FILE *fp;
      int count=0;
      char ch, *fname;
      clrscr();

      printf("\n Enter the file name:");
      scanf("%s", fname);


      fp=fopen (fname,"r");

      if (fp == NULL)
      {
            printf("\n Error opening the file.");
            exit(0);
      }



      while(  (ch=fgetc(fp)) != EOF)
      {
      count++;
      printf("\n CHAR READ FORM FILE = %c",ch);
      //putchar(ch);
      }
```

```
    printf("\n Number of chars in the file: %d", count);
    getch();
}
```

OUTPUT:

```
Enter the file name: abhi.txt
CHAR READ FORM FILE = I
 CHAR READ FORM FILE =
 CHAR READ FORM FILE = a
 CHAR READ FORM FILE = m
 CHAR READ FORM FILE =
 CHAR READ FORM FILE = a
 CHAR READ FORM FILE =
 CHAR READ FORM FILE = G
 CHAR READ FORM FILE = P
 CHAR READ FORM FILE = T
 CHAR READ FORM FILE =
 CHAR READ FORM FILE = G
 CHAR READ FORM FILE = a
 CHAR READ FORM FILE = d
 CHAR READ FORM FILE = a
 CHAR READ FORM FILE = g
 CHAR READ FORM FILE =
 CHAR READ FORM FILE = S
 CHAR READ FORM FILE = t
 CHAR READ FORM FILE = u
 CHAR READ FORM FILE = d
 CHAR READ FORM FILE = e
 CHAR READ FORM FILE = n
 CHAR READ FORM FILE = t
 Number of chars in the file: 24
```

7 . WRITE A C PROGRAM TO CREATE A FILE THAT CONTAINS AT LEAST 5 RECORDS WHICH CONSISTS OF BOOK NO., BOOK NAME, AUTHOR, PUBLISHER, AND PRICE.

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

struct book
{
      int bno;
      char bname[50];
      char author[50];
      char pub[50];
      int price;
};


int main()
{
      FILE *fp ;
      struct book ba[5];
      char fname[50] ;
      int i, n ;
      clrscr() ;

      printf("\nEnter the filename: ");
      scanf("%s", fname);

      fp = fopen(fname, "w") ;

      if (fp == NULL)
      {
            printf("\nError creating the file.");
```

```
            exit(0);
      }

      printf("\nHow many records to be written to file(max 5): ");
      scanf("%d", &n);

      for(i=0; i < n; i++)
      {
            printf("\n Enter BookNo, BookName, Author, Publ,
Price:\n");
            scanf("%d%s%s%s%d", &ba[i].bno, ba[i].bname,
ba[i].author, ba[i].pub, &ba[i].price );
            fprintf(fp, "%d\t%s\t%s\t%s\t%d\n", ba[i].bno,
ba[i].bname, ba[i].author, ba[i].pub, ba[i].price );

      }
      printf("\nFile is written: %s", fname);

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

OUTPUT :

```
Enter the filename: abhi.txt

How many records to be written to file(max 5): 2

 Enter BookNo, BookName, Author, Publ, Price:
103 DataStructures SatishHongal EBPB 350

 Enter BookNo, BookName, Author, Publ, Price:
107 Java Abhisheksingh Delhi 400

File is written: abhi.txt
```

8 . WRITE A C PROGRAM TO DISPLAY THE CONTENTS OF THE FILE CREATED IN PROGRAM NO. 5 IN THE FOLLOWING FORMAT BOOK NO. BOOK NAME AUTHOR PUBLISHER PRICE.

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

void main()
{
      FILE *fp;
      char ch;
      char fname[50];
      clrscr();
      printf("\nEnter the file name to be read: ");
      scanf("%s", fname);
      fp = fopen(fname, "r");
      if(fp == NULL)
      {
            printf("\nCannot open the file: %s",fname);
            exit(0);
      }
      printf("\nBookNo\tBName\tAuthor\tPubl\tPrice\n");
      while((ch=fgetc(fp)) != EOF)
      {
            putchar(ch);
      }
      getch();
}
```

OUTPUT:

```
Enter the file name to be read: abhi.txt

BookNo  BName   Author  Publ    Price
103     DataStructures  SatishHongal    EBPB    350
107     Java    Abhisheksingh    Delhi   400
```

9 . WRITE A C PROGRAM TO COPY ONE FILE TO ANOTHER FILE USING COMMAND LINE ARGUMENTS.

```
#include<stdio.h>
#include<conio.h>
void main(int argc,char *agrv[])
{
      FILE *fp1,*fp2;
      char ch;
      clrscr();
      if (argc !=3)
{
      printf("\n invalid input");
      exit(0);
}
      fp1=fopen(argc[1];"r");
      fp2=fopen(argc[2];"w");
      if(fp1 == NULL || fp2 == NULL )
{
      printf("\n file ncant be opened / created ");
      exit(0);
}
      while((ch=fgetc(fp1))!=EOF)
        fput(ch,fp2);
        fclose(fp1);
        fclose(fp2);

      printf("\n file copied succesfully");
      getch();

}
```

OUTPUT:

```
Abhi01.txt Abhi02.txt
file copied succesfully
```

10. WRITE A C PROGRAM TO IMPLEMENT SINGLY LINKED LIST: INSERT, DELETE, SEARCH AND DISPLAY.

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

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

typedef struct node NODE ;

NODE* getnewnode()
{
      NODE* newnode = (NODE*)malloc(sizeof(struct node));
      printf("\nEnter the integer data to be inserted: ");
      scanf("%d", &newnode->data);
      newnode->next = NULL;

      return newnode ;
}


NODE* insert_front(NODE* start)
{
      NODE* newnode = getnewnode();

      newnode->next = start ;


      return newnode;
}
```

```
NODE* delete_front(NODE* start)
{
      NODE* temp = start ;

      if(start == NULL)
      {
            printf("\nLinked List is EMPTY!");
            return start;
      }

      start = start->next ;

      printf("\nThe deleted element: %d", temp->data);
      free(temp);

      return start;

}

void display(NODE* start)
{
      NODE* temp = start;
      if(start == NULL)
      {
            printf("Linked List is EMPTY!");
            return;
      }
      printf("\nLinked List Contents:\n");
      while(temp != NULL)
      {
```

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

void search(NODE* start)
{
      int key ;
      NODE* temp=start;

      if(start == NULL)
      {
            printf("\nLIST IS EMPTY");
            return;
      }

      printf("\nEnter the data to be searched: ");
      scanf("%d", &key);

      while(temp != NULL)
      {
            if(temp->data == key)
            {
                  printf("\nThe data %d is found!", key);
                  return;
            }
            temp = temp->next;
      }
      printf("\nData %d is not found", key);
```

```
}

void main()
{
      NODE *start=NULL ;
      int ch;
      clrscr();

      while(1)
      {
            printf("\n 1:Insert_FRONT 2:Delete_FRONT 3:Display
4:Search 5:Exit");
            printf("\n Enter the choice: ");
            scanf("%d", &ch);

            switch(ch)
            {
                  case 1: start = insert_front(start) ;
                        break;

                  case 2: start = delete_front(start) ;
                        break;
                  case 3:
                        display(start);
                        break;
                  case 4: search(start);
                        break;
                  default:
                         exit(0);
            }
      }
}
```

OUTPUT:

```
1:Insert_FRONT 2:Delete_FRONT 3:Display 4:Search 5:Exit
 Enter the choice: 1

Enter the integer data to be inserted: 10

 1:Insert_FRONT 2:Delete_FRONT 3:Display 4:Search 5:Exit
 Enter the choice: 1

Enter the integer data to be inserted: 20

 1:Insert_FRONT 2:Delete_FRONT 3:Display 4:Search 5:Exit
 Enter the choice: 1

Enter the integer data to be inserted: 30

 1:Insert_FRONT 2:Delete_FRONT 3:Display 4:Search 5:Exit
 Enter the choice: 1

Enter the integer data to be inserted: 40

 1:Insert_FRONT 2:Delete_FRONT 3:Display 4:Search 5:Exit
 Enter the choice: 3

Linked List Contents:
40      30      20      10
 1:Insert_FRONT 2:Delete_FRONT 3:Display 4:Search 5:Exit
 Enter the choice: 2

The deleted element: 40
 1:Insert_FRONT 2:Delete_FRONT 3:Display 4:Search 5:Exit
 Enter the choice: 2
```

```
The deleted element: 30
 1:Insert_FRONT 2:Delete_FRONT 3:Display 4:Search 5:Exit
 Enter the choice: 3

Linked List Contents:
20      10
 1:Insert_FRONT 2:Delete_FRONT 3:Display 4:Search 5:Exit
 Enter the choice: 4

Enter the data to be searched: 10

The data 10 is found!
 1:Insert_FRONT 2:Delete_FRONT 3:Display 4:Search 5:Exit
 Enter the choice: 5
```

11. WRITE A C PROGRAM TO ILLUSTRATE STACK OPERATIONS USING ARRAYS.

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

#define STACK_SIZE 5

struct stack
{
      int element[STACK_SIZE];
      int top;
};
      struct stack s;
      void push()
{
      int data;
      if(s.top == STACK_SIZE -1)
{
      printf("\nStack overflow!");
      return;
}

      printf("\n Enter the data to be pushed on to stack:");
      scanf("%d", &data);
      s.top++;
      s.element[s.top] = data;
      printf("\n%d is pushed on to the stack.",data);
      return;
      }

      void pop()
```

```
{
if(s.top == -1)
{
printf("\n stack underflow!");
return;
}
printf("\n the popped out element is %d", s.element[s.top]);
s.top--;
return;
}
void display()
{
int i;
if(s.top == -1)
{
printf("\n stack is empty!");
return;
}
printf("\n stack contents:\n");
for(i=0; i<=s.top; i++)
printf("\t%d", s.element[i]);
}
int main()
{
int ch;
s.top++;
clrscr();
while(1)
{
printf("\n STACK OPERATION :1: PUSH 2: POP 3:Display 4:exit");
printf("\n enter the choice:");
scanf("%d", &ch);
```

```
      switch(ch)
      {
      case 1: push();
            break;
      case 2: pop();
            break;
      case 3: display();
                  break;
      default:
      exit(0);

      return 0;
}
}
}
```

OUTPUT:

```
STACK OPERATION :1: PUSH 2: POP 3:Display 4:exit
 enter the choice:1

 Enter the data to be pushed on to stack:20

20 is pushed on to the stack.
 STACK OPERATION :1: PUSH 2: POP 3:Display 4:exit
 enter the choice:1

 Enter the data to be pushed on to stack:30

30 is pushed on to the stack.
 STACK OPERATION :1: PUSH 2: POP 3:Display 4:exit
 enter the choice:1
```

```
 Enter the data to be pushed on to stack:40

40 is pushed on to the stack.
 STACK OPERATION :1: PUSH 2: POP 3:Display 4:exit
 enter the choice:3

 stack contents:
        0       0       20      30      40
 STACK OPERATION :1: PUSH 2: POP 3:Display 4:exit
 enter the choice:2

 the popped out element is 40
 STACK OPERATION :1: PUSH 2: POP 3:Display 4:exit
 enter the choice:2

 the popped out element is 30
 STACK OPERATION :1: PUSH 2: POP 3:Display 4:exit
 enter the choice:2

 the popped out element is 20
 STACK OPERATION :1: PUSH 2: POP 3:Display 4:exit
 enter the choice:4
```

12. WRITE A C PROGRAM TO FIND THE GCD OF TWO NUMBERS USING RECURSION.

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

int gcd(int x, int y)
{
      if(x != y)
      {
            if(x > y)
                  return gcd(x-y, y);
            else
                 return gcd(x, y-x);
      }
      return x ;
}
int main()
{
      int x, y, res ;
      clrscr();

      printf("\nEnter two numbers: ");
      scanf("%d%d", &x, &y);

      res = gcd(x, y);
      printf("\n The GCD of %d & %d = %d", x, y, res);
      getch();
}
```

OUTPUT:

```
Enter two numbers: 20 45

 The GCD of 20 & 45 = 5
```

14. WRITE A C PROGRAM TO IMPLEMENT QUEUE USING ARRAYS.

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#define SIZE 3

struct queue
{
      int items[SIZE];
      int front,rear;
};
      struct queue q;
      void qInsert()
      {
      int data;
      if(q.rear == SIZE-1)
      {
      printf("\nQueue overflow!");
      printf("\n Front=%d Rear=%d",q.front,q.rear);
      return;
      }

      printf("\n Enter the data to be inserted into the Queue");
      scanf("%d", &data);
      q.rear++;
      q.items[q.rear] = data;
      if(q.front == -1)
      q.front++;

      printf("\n%d is insert into the queue.",data);
      printf("\n Front=%d Rear=%d", q.front,q.rear);
      return;
```

```
}

 void qDelete()
 {
      if(q.front == -1)
 {

 printf("\nQueue underflow!");
 return;
 }
 printf("\nBefore Delete Front=%d Rear=%d",q.front,q.rear);
 printf("\n The deleted item is %d",q.items[q.front]);
 if(q.front == q.rear)
 q.front = q.rear = -1;

 else
 q.front++;
 printf("\nBefore Delete Front=%d Rear=%d", q.front,q.rear);
 }

 void qDisplay()
{
int i;
if(q.front == -1)
{
printf("\nQueue is empty!");
return;
}
printf("\nQueue Contents:\n");
for(i=q.front; i<=q.rear; i++)
printf("\t%d", q.items[i]);
}
```

```
int main()
{
 int ch;
 q.front = q.rear = -1;
 clrscr();

 printf("\nFront=%d Rear=%d", q.front,q.rear);

 while(1)
 {

 printf("\n QUEUE OPERATION : 1:QInsert 2:QDelete 3:Display
4:Exit");
 printf("\n enter the choice:");
 scanf("%d", &ch);

 switch(ch)
 {

 case 1: qInsert();
 break;
 case 2: qDelete();
 break;
 case 3: qDisplay();
 break;

 default:
 exit(0);

 return 0;
 }
 }
 }
```

OUTPUT:

```
Front=-1 Rear=-1
 QUEUE OPERATION : 1:QInsert 2:QDelete 3:Display 4:Exit
 enter the choice:1

 Enter the data to be inserted into the Queue 30

30 is insert into the queue.
 Front=0 Rear=0
 QUEUE OPERATION : 1:QInsert 2:QDelete 3:Display 4:Exit
 enter the choice: 1

 Enter the data to be inserted into the Queue 60

60 is insert into the queue.
 Front=0 Rear=1
 QUEUE OPERATION : 1:QInsert 2:QDelete 3:Display 4:Exit
 enter the choice: 1

 Enter the data to be inserted into the Queue 78

78 is insert into the queue.
 Front=0 Rear=2
 QUEUE OPERATION : 1:QInsert 2:QDelete 3:Display 4:Exit
 enter the choice: 3

Queue Contents:
        30      60      78
 QUEUE OPERATION : 1:QInsert 2:QDelete 3:Display 4:Exit
 enter the choice:
2
```

```
Before Delete Front=0 Rear=2
 The deleted item is 30
Before Delete Front=1 Rear=2
 QUEUE OPERATION : 1:QInsert 2:QDelete 3:Display 4:Exit
 enter the choice:2

Before Delete Front=1 Rear=2
 The deleted item is 60
Before Delete Front=2 Rear=2
 QUEUE OPERATION : 1:QInsert 2:QDelete 3:Display 4:Exit
 enter the choice:2

Before Delete Front=2 Rear=2
 The deleted item is 78
```

16. WRITE A C PROGRAM TO IMPLEMENT BINARY TREE TRAVERSAL OPERATIONS.

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
struct node *left;
int data;
struct node *right;
};

typedef struct node* NODE;

void insert(NODE *p,int num) {
      if((*p)==NULL)
      {
            printf("Leaf node created.");
            (*p) = (struct node*) malloc(sizeof(struct node));
            (*p)->left = NULL;
            (*p)->right = NULL;
            (*p)->data = num;
            return;
      }
      else
      {
            if(num == (*p)->data)
            {
                  printf("\nREPEATED ENTRY ERROR VALUE REJECTED\n");
                  return;
            }

            if(num < (*p)->data)
            {
```

```
                printf("\nDirected to left link.\n");
                insert(&((*p)->left), num);
            }
            else
            {
                printf("Directed to right link.\n");
                insert(&((*p)->right), num);
            }
        }
        return ;
}

void inorder(NODE p)
{
        if(p != NULL)
        {
            inorder(p->left);
            printf("%d\t", p->data);
            inorder(p->right);
        }
        else
            return ;
}

void preorder(NODE p)
{
        if(p != NULL)
        {
            printf("%d\t", p->data);
            preorder(p->left);
            preorder(p->right);
        }
```

```
      else
            return ;
}

void postorder(NODE p)
{
      if(p != NULL)
      {
            postorder(p->left);
            postorder(p->right);
            printf("%d\t", p->data);
      }
      else
            return ;
}

int main()
{
      NODE bt = NULL;
      int i, n, item;
      printf("\nProgram for Tree Traversal\n");
      printf("Enter the number of nodes to add to the binary
tree.\n");
      scanf("%d",&n);

      for(i=0;i<n;i++)
      {
            printf("\nEnter the item\n");
            scanf("%d",&item);
            insert(&bt,item);
      }

      printf("\nInorder Traversal\n");
```

```
      inorder(bt);

      printf("\nPreorder Traversal\n");
      preorder(bt);

      printf("\nPostorder Traversal\n");
      postorder(bt);
      printf("\n");
      return 0;
}
```

OUTPUT:

```
Program for Tree Traversal
Enter the number of nodes to add to the binary tree.
4

Enter the item
12
Leaf node created.
Enter the item
33
Directed to right link.
Leaf node created.
Enter the item
56
Directed to right link.
Directed to right link.
Leaf node created.
Enter the item
89
Directed to right link.
Directed to right link.
```

```
Directed to right link.
Leaf node created.
Inorder Traversal
12      33      56      89
Preorder Traversal
12      33      56      89
Postorder Traversal
89      56      33      12
```

17. WRITE A C PROGRAM TO SORT AN ARRAY USING BUBBLE SORT.

```
#include<stdio.h>

void bubblesort(int a[],int n)
{
      int i, j, temp;

      for(i=0; i < n-1 ; i++)// controls the no. of passes
      {
            for(j=0; j<n-i-1 ; j++) // to compare & interchange
elements for ascending order
            {
                  if(a[j] > a[j+1]) // elements out of order an
exchange is necessary
                  {
                        temp = a[j];
                        a[j] = a[j+1];
                        a[j+1] = temp;
                  }
            }
      }
}
```

```
int main()
{
      int i, n, a[20];

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

      printf("\n Enter the elements of the array to be sort: ");

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

      bubblesort(a,n);

      printf("The sorted elements of the array are: \n");

      for(i=0;i<n;i++)
            printf("%d\t",a[i]);

      printf("\n");
      return 0;
}
```

OUTPUT :

```
Enter the number of elements in the array: 5

 Enter the elements of the array to be sort: 89 34 78 36 11
The sorted elements of the array are:
11      34      36      78      89
```

18. WRITE A C PROGRAM TO SORT AN ARRAY USING SELECTION SORT.

```
#include<stdio.h>

void selectionsort(int a[],int n)
{
      int i, j, pos, small, temp;

      for(i=0;i<n-1;i++) //no of passes
      {
            small = a[i];
            pos = i;
            for(j = i+1; j < n ; j++) // no of checking
            {
                  if(a[j] < small)
                  {
                        small = a[j];
                        pos = j;
                  }
            }

            temp = a[pos];
            a[pos] = a[i];
            a[i] = temp;

      }
}

int main()
{
      int i, n, a[20];
```

```
	printf("Enter the number of elements to sort: ");
	scanf("%d",&n);

	printf("\n Enter the elements to sort: ");
	for(i=0 ; i < n; i++)
		scanf("%d", &a[i]);

	selectionsort(a, n);
	printf("The sorted elements are: \n");
	for(i=0;i<n;i++)
		printf("%d\t",a[i]);

	printf("\n");
	//getch();
	return 0;
}
```

OUTPUT :

```
Enter the number of elements to sort:  5

 Enter the elements to sort: 45 13 78 64 99
The sorted elements are:
13      45      64      78      99
```

19. WRITE A C PROGRAM TO SEARCH A GIVEN NUMBER USING LINEAR SEARCH.

```
#include<stdio.h>

int main()
{
      int a[20], size, key, flag, i ;

      printf("Enter the size of an array: \n");
      scanf("%d", &size);

      printf("Enter %d elements into the array: \n",size);

      for(i=0;i<size;i++)
      {
            scanf("%d",&a[i]);
      }
      printf("Enter the element to be searched:\n");
      scanf("%d",&key);
      flag = 0;

      for(i=0;i<size;i++)
      {
            if(key == a[i])
            {
                  flag = 1;
                  break;
            }
      }
      if(flag==1)
            printf("Search successful at position %d\n",i+1);
      else
```

```
            printf("Search Unsuccessful\n");


      return 0;
}
```

OUTPUT:

```
Enter the size of an array:
5
Enter 5 elements into the array:
56 78 34 90 33
Enter the element to be searched:
34
Search successful at position 3
```

20. WRITE A C PROGRAM TO SEARCH A GIVEN NUMBER USING BINARY SEARCH.

```
#include<stdio.h>

int binarysearch(int array[], int key, int size)
{
      int start, end, mid;
      start = 0;
      end = size-1;

      while(start<=end)
      {
            mid = (start + end)/2;

            if(key == array[mid])
                  return (mid + 1);

            if(key < array[mid])
                  end = mid - 1 ;
            else
                  start = mid + 1 ;
      }
      return 0;
}

int main()
{
      int i, n, key, location, array[50];
      printf("\n Enter the number of elements: ");
      scanf("%d", &n);
      printf("\n Enter the array items: \n");

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

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

      printf("\n Enter the search element: ");
      scanf("%d",&key);

      location = binarysearch(array,key,n);

      if(location == 0)
            printf("\n Element not found and search unsuccessful
\n");
      else
            printf("\n Search Successful and item found at %d
location \n",location);

      return 0;
}
```

OUTPUT:

```
Enter the number of elements: 3

 Enter the array items:
21 44 55

 Enter the search element: 44

 Search Successful and item found at 2 location
```