LINEAR SEARCH

SOURCE CODE

```
#include<stdio.h>
#include<conio.h>
void main()
{
int i,n,a[20],x,flag=0;
clrscr();
printf("\nlinear search:");
printf("\n\nEnter the number of element:\n");
scanf("%d",&n);
printf("\nEnter the array element:\n");
for(i=0;i<n;i++)
{
scanf("%d",&a[i]);
}
printf("\nEnter the element to be search\n");
scanf("%d",&x);
for(i=0;i<n;i++)
{
if(x==a[i])
{
printf("\n The given element position %d is found in    position %d\n",x,i+1);
flag=1;
break;
}
}
if(flag==0)
{
printf("\n Element not found\n");
}
getch();
}
```

OUTPUT

```
 linear search

  Enter The Number of an element
5

  Enter the array element:
34
56
98
12
11

  Enter the element to be searched
11

  the given element its position 11 is found in position 5
```

```
linear search

 Enter The Number of an element
5

 Enter the array element:
34
56
67
89
23

 Enter the element to be searched
99

 Element not found
_
```

BINARY SEARCH

SOURCE CODE

```
#include <stdio.h>
#include<conio.h>
int main()
clrscr();
{
int c, first, last, middle, n, search, array[100];
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;
}
```

OUTPUT

```
Enter number of elements
4
Enter 4 integers
55
67
78
23
Enter value to find
78
78 found at location 3.

Process returned 0 (0x0)   execution time : 65.561 s
Press any key to continue.
```

```
Enter number of elements
5
Enter 5 integers
33
6
08
09
02
Enter value to find
99
Not found! 99 isn't present in the list.

Process returned 0 (0x0)   execution time : 20.014 s
Press any key to continue.
```

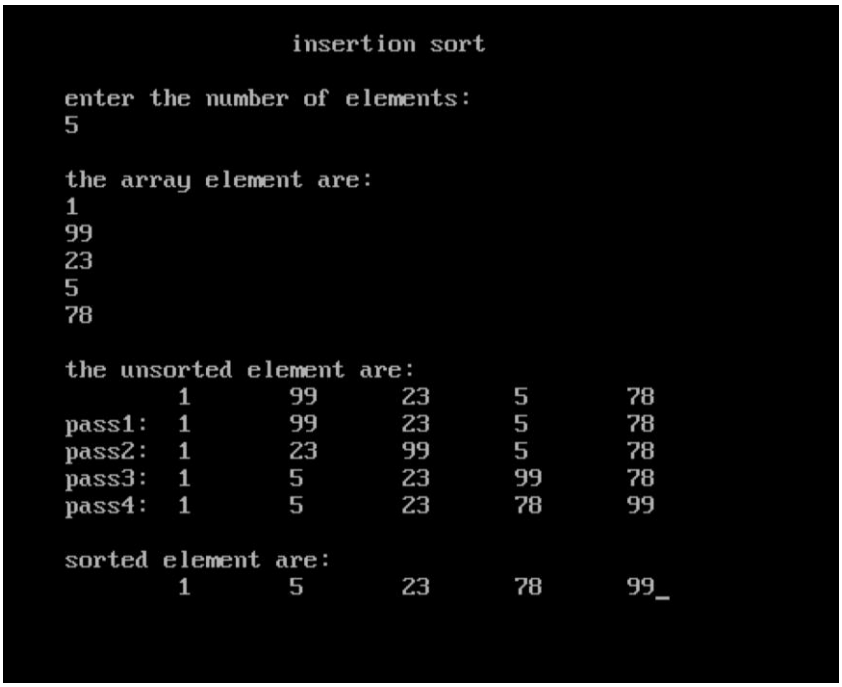
INSERTION SORT

SOURCE CODE

```
#include<stdio.h>
#include<conio.h>
int a[20],i,j,k,n,key;
void get();
void sort();
void display(int[],int);
void main()
{
clrscr();
printf("\n\t\tinsertion sort\n");
get();
sort();
getch();
}
void get()
{
printf("\nenter the number of elements:\n");
scanf("%d",&n);
printf("\nthe array element are:\n");
for(i=1;i<=n;i++)
{
scanf("%d",&a[i]);
}
printf("\nthe unsorted element are:\n");
display(a,n);
}
void sort()
{
for(j=2;j<=n;j++)
{
key=a[j];
i=j-1;
while((i>0)&&(key<a[i]))
{
a[i+1]=a[i];
i=i-1;
}
a[i+1]=key;
printf("\npass%d:",j-1);
display(a,n);
}
```

```
printf("\n\nsorted element are:\n");
display(a,n);
}
void display(int a[],int n)
{
for(i=1;i<=n;i++)
{
printf("\t%d",a[i]);
}
}
```

OUTPUT


insertion sort
enter the number of elements:
5
the array element are:
1
99
23
5
78
the unsorted element are:
1 99 23 5 78
pass1: 1 99 23 5 78
pass2: 1 23 99 5 78
pass3: 1 5 23 99 78
pass4: 1 5 23 78 99
sorted element are:
1 5 23 78 99

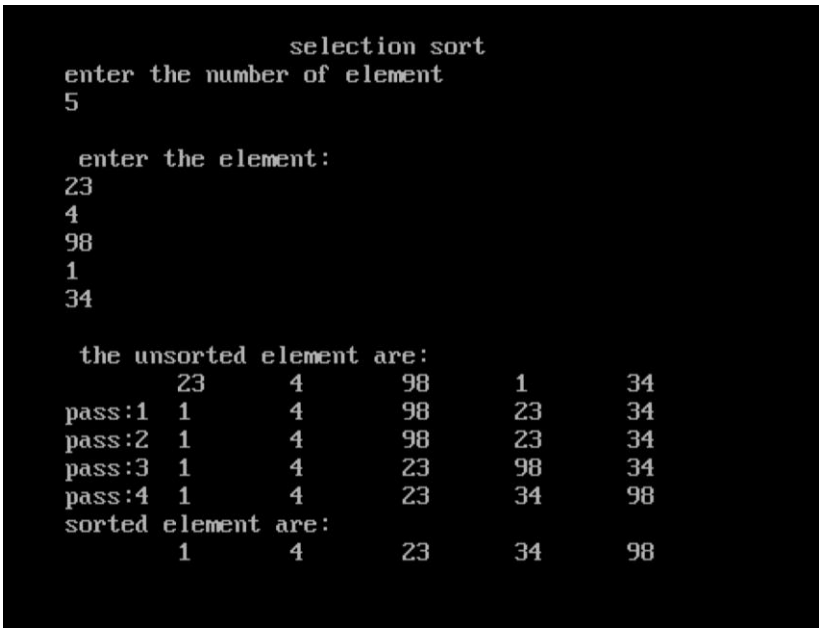
SELECTION SORT

SOURCE CODE

```
#include<stdio.h>
#include<conio.h>
int i,j,n,k,a[20],min;
void get();
void sort();
void display(int[],int);
void main()
{
clrscr();
printf("\n\t\tselection sort");
get();
sort();
getch();
}
void get()
{
printf("\nenter the number of element \n");
scanf("%d",&n);
printf("\n enter the element:\n");
for(i=1;i<=n;i++)
{
scanf("%d",&a[i]);
}
printf("\n the unsorted element are:\n");
display(a,n);
}
void sort()
{
for(k=1;k<n;k++)
{
min=k;
for(j=k;j<=n;j++)
{
if(a[j]<a[min])
{
min=j;
}
}
if(min!=k)
{
t=a[k];
a[k]=a[min];
```

```
a[min]=t;
}
printf("\npass:%d",k);
display(a,n);
}
printf("\nsorted element are:\n");
display(a,n);
}
void display(int a[],int n)
{
for(i=1;i<=n;i++)
{
printf("\t%d",a[i]);
}
}
```

OUTPUT

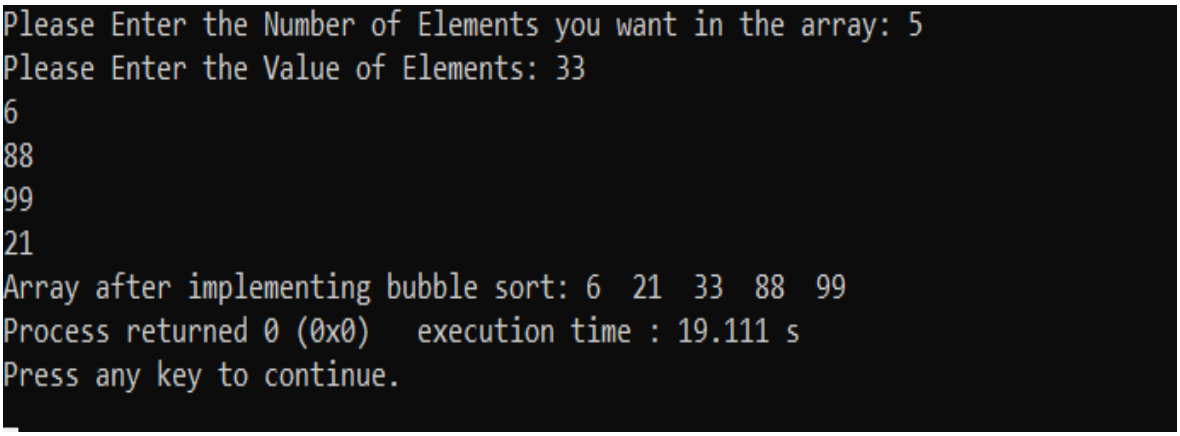

selection sort
enter the number of element
5
enter the element:
23
4
98
1
34
the unsorted element are:
23 4 98 1 34
pass:1 1 4 98 23 34
pass:2 1 4 98 23 34
pass:3 1 4 23 98 34
pass:4 1 4 23 34 98
sorted element are:
1 4 23 34 98

BUBBLE SORT

SOURCE CODE

```
#include <stdio.h>
#include<conio.h>
int main(){
int arr[50], num, x, y, temp;
printf("Please Enter the Number of Elements you want in the array: ");
scanf("%d", &num);
printf("Please Enter the Value of Elements: ");
for(x = 0; x < num; x++)
scanf("%d", &arr[x]);
for(x = 0; x < num - 1; x++){
for(y = 0; y < num - x - 1; y++){
if(arr[y] > arr[y + 1]){
temp = arr[y];
arr[y] = arr[y + 1];
arr[y + 1] = temp;
}
}
}
printf("Array after implementing bubble sort: ");
for(x = 0; x < num; x++){
printf("%d  ", arr[x]);
}
return 0;
}
```

OUTPUT


Please Enter the Number of Elements you want in the array: 5
Please Enter the Value of Elements: 33
6
88
99
21
Array after implementing bubble sort: 6 21 33 88 99
Process returned 0 (0x0) execution time : 19.111 s
Press any key to continue.

SHELL SORT

SOURCE CODE

```
#include <stdio.h>
#include<conio.h>
void shellsort(int arr[], int num)
{
int i, j, k, tmp;
for (i = num / 2; i > 0; i = i / 2)
{
for (j = i; j < num; j++)
{
for(k = j - i; k >= 0; k = k - i)
{
if (arr[k+i] >= arr[k])
break;
else
{
tmp = arr[k];
arr[k] = arr[k+i];
arr[k+i] = tmp;
}
}
}
}
}
int main()
{
int arr[30];
int k,  num;
printf("Enter total no. of elements : ");
scanf("%d", &num);
printf("\nEnter %d numbers: ", num);
for (k =  0 ; k < num; k++)
{
scanf("%d", &arr[k]);
}
shellsort(arr, num);
printf("\n Sorted array is: ");
for (k = 0; k < num; k++)
printf("%d ", arr[k]);
return 0;
}
```

OUTPUT

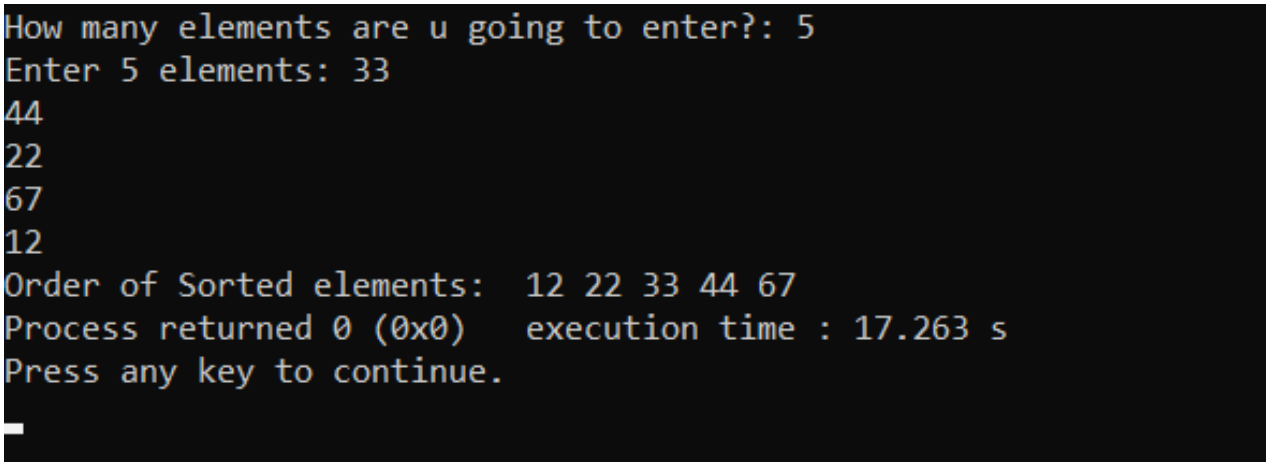
```
Enter total no. of elements : 6

Enter 6 numbers: 44
5
66
8
123
456

 Sorted array is: 5 8 44 66 123 456
Process returned 0 (0x0)   execution time : 14.674 s
Press any key to continue.
```

QUICK SORT

SOURCE CODE

```
#include<stdio.h>
#include<conio.h>
void quicksort(int number[25],int first,int last){
int i, j, pivot, temp;
if(first<last){
pivot=first;
i=first;
j=last;
while(i<j){
while(number[i]<=number[pivot]&&i<last)
i++;
while(number[j]>number[pivot])
j--;
if(i<j){
temp=number[i];
number[i]=number[j];
number[j]=temp;
}
}
temp=number[pivot];
number[pivot]=number[j];
number[j]=temp;
quicksort(number,first,j-1);
quicksort(number,j+1,last);
}
}
int main()
{
int i, count, number[25];
printf("How many elements are u going to enter?: ");
scanf("%d",&count);
printf("Enter %d elements: ", count);
for(i=0;i<count;i++)
scanf("%d",&number[i]);
quicksort(number,0,count-1);
printf("Order of Sorted elements: ");
for(i=0;i<count;i++)
printf(" %d",number[i]);
return 0;
}
```

OUTPUT


How many elements are u going to enter?: 5
Enter 5 elements: 33
44
22
67
12
Order of Sorted elements: 12 22 33 44 67
Process returned 0 (0x0)   execution time : 17.263 s
Press any key to continue.

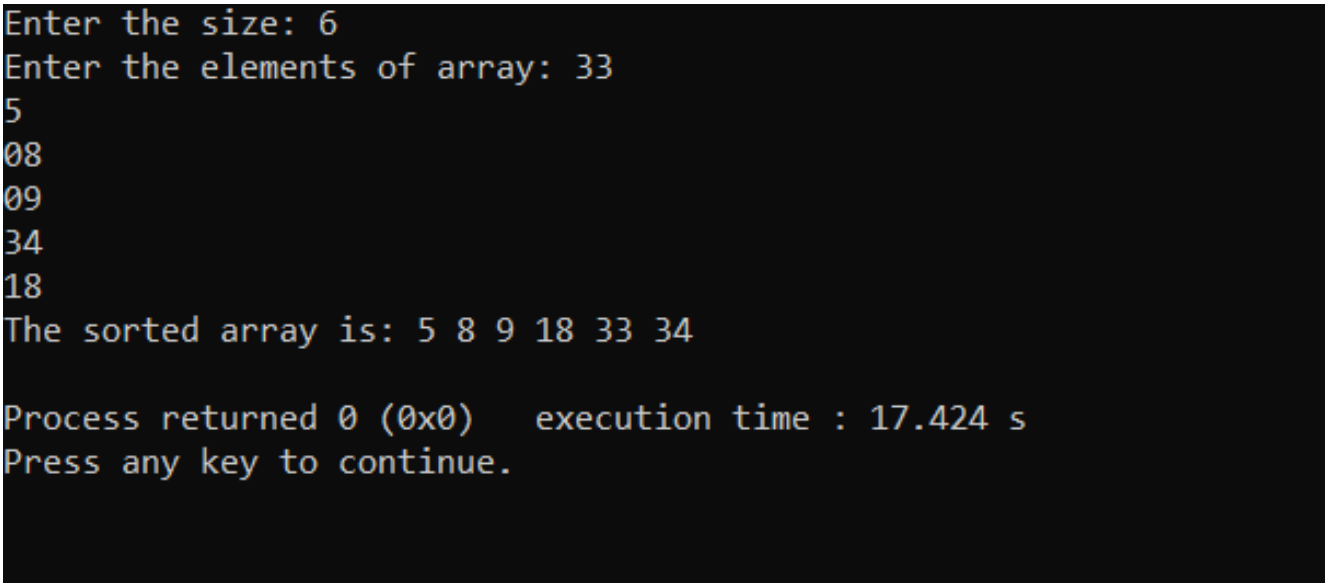
MERGE SORT

SOURCE CODE

```
#include <stdio.h>
#include<conio.h>
// merge function
void Merge(int arr[], int left, int mid, int right)
{
int i, j, k;
int size1 = mid - left + 1;
int size2 = right - mid;
// created temporary array
int Left[size1], Right[size2];
// copying the data from arr to temporary array
for (i = 0; i < size1; i++)
Left[i] = arr[left + i];
for (j = 0; j < size2; j++)
Right[j] = arr[mid + 1 + j];
// merging of the array
i = 0;     // intital index of first subarray
j = 0;     // inital index of second subarray
k = left; // initial index of parent array
while (i < size1 && j < size2)
{
if (Left[i] <= Right[j])
{
arr[k] = Left[i];
i++;
}
Else
{
arr[k] = Right[j];
j++;
}
k++;
}
// copying the elements from Left[], if any
while (i < size1)
{
arr[k] = Left[i];
i++;
k++;
}
// copying the elements from Right[], if any
```

```
while (j < size2)
{
arr[k] = Right[j];
j++;
k++;
}
}
//merge sort function
void Merge_Sort(int arr[], int left, int right)
{
if (left < right)
{
int mid = left + (right - left) / 2;
// recursive calling of merge_sort
Merge_Sort(arr, left, mid);
Merge_Sort(arr, mid + 1, right);
Merge(arr, left, mid, right);
}
}
// driver code
int main()
{
int size;
printf("Enter the size: ");
scanf("%d", &size);
int arr[size];
printf("Enter the elements of array: ");
for (int i = 0; i < size; i++)
{
scanf("%d", &arr[i]);
}
Merge_Sort(arr, 0, size - 1);
printf("The sorted array is: ");
for (int i = 0; i < size; i++)
{
printf("%d ", arr[i]);
}
printf("\n");
return 0;
}
```

OUTPUT


Enter the size: 6
Enter the elements of array: 33
5
08
09
34
18
The sorted array is: 5 8 9 18 33 34
Process returned 0 (0x0)   execution time : 17.424 s
Press any key to continue.

RADIX SORT

SOURCE CODE

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
int *Radix_sort(int *arr, int size);
int *Count_sort(int *arr, int size, int Exponent);
int maximum(int *arr,int length);
int minimum(int *arr,int length);
int main()
{
int i, size;
int *arr;
printf("Enter the array size: ");
scanf("%d",&size);
arr = (int *) malloc( sizeof( int ) * size );
if(arr==NULL)
{
exit(-1);//abnormal termination.
}
else
{
// Entering the array values
printf("Enter the array\n");
for(i = 0; i < size; i++)
{
printf("arr[ %d ] = ",i);
scanf("%d",&arr[i]);
}
printf("Array before sorting:\n");
for(i = 0; i < size; i++)
{
printf("arr[%d] = %d\n",i,arr[i]);
}
arr = Radix_sort(arr,size);
}
printf("ARRAY AFTER SORTING: \n");
for(int i=0;i<size;i++)
{
printf("arr[ %d ] = %d \n",i ,arr[i]);
}
}
int *Radix_sort(int *arr, int size)
```

```
{
int Max_of_arr = maximum(arr, size);
int Exponent = 1;
int count = 0;
while(Exponent <= Max_of_arr)
{
arr = Count_sort(arr, size, Exponent);
Exponent= Exponent* 10;
//uncomment the loop to see how sorting happens digit after moving from LSB to MSB.
/*
printf("ARRAY AFTER SORTING: %d digits from rightmost element\n",count);
for(int i=0;i<size;i++)
printf("arr[ %d ] = %d \n",i ,arr[i]);
count++;
*/
}
return arr;
}
int *Count_sort(int *arr, int size, int Exponent)
{
int range = 10;
int *frequency_array ;
frequency_array = (int*)malloc(sizeof(int)* range);
if(frequency_array == NULL)
{
exit(-1);
}
int sum=0;
for(int i=0; i<range; i++)
{
frequency_array[ i ]=0;
}
for(int i=0;i<size;i++)
{
frequency_array[ (arr[i]/Exponent)%10 ]++;
}
for(int i =0; i<range;i++)
{
sum = sum + frequency_array[i];
frequency_array[i] = sum;
}
int *new_arr;//new array to store the result.
new_arr = (int *)malloc(sizeof(int) *size);
if(new_arr == NULL)
{
exit(-1);
}
```

```
else
{
int pos;
for(int i=size-1; i>=0 ;i-- )
{
pos = frequency_array[(arr[i]/Exponent)%10]-1;
new_arr[ pos ] = arr[ i ];
frequency_array [(arr[i]/Exponent)%10]--;
}
}
return new_arr;
}
int maximum(int *arr, int length)
{
int max=INT_MIN;
for( int i=0 ; i<length ; i++ )
{
if(arr[i]>max)
max=arr[i];
}
return max;
}
int minimum(int *arr, int length)
{
int min=INT_MAX;
for( int i=0 ; i<length ; i++ )
{
if(arr[i]<min)
min=arr[i];
}
return min;
}
```

OUTPUT

```
Enter the array size: 7
Enter the array
arr[ 0 ] = 234
arr[ 1 ] = 2
arr[ 2 ] = 456
arr[ 3 ] = 789
arr[ 4 ] = 23
arr[ 5 ] = 56
arr[ 6 ] = 89
Array before sorting:
arr[0] = 234
arr[1] = 2
arr[2] = 456
arr[3] = 789
arr[4] = 23
arr[5] = 56
arr[6] = 89
ARRAY AFTER SORTING:
arr[ 0 ] = 2
arr[ 1 ] = 23
arr[ 2 ] = 56
arr[ 3 ] = 89
arr[ 4 ] = 234
arr[ 5 ] = 456
arr[ 6 ] = 789

Process returned 0 (0x0)   execution time : 30.637 s
Press any key to continue.
```

SINGLE LINKED LIST

SOURCE CODE

```
#include <stdio.h>
#include<conio.h>
#include <malloc.h>
#include <stdlib.h>
struct node {
int value;
struct node *next;
};
void insert();
void display();
void delete();
int count();
typedef struct node DATA_NODE;
DATA_NODE *head_node, *first_node, *temp_node = 0, *prev_node, next_node;
int data;
int main() {
int option = 0;
printf("Singly Linked List Example - All Operations\n");
while (option < 5) {
printf("\nOptions\n");
printf("1 : Insert into Linked List \n");
printf("2 : Delete from Linked List \n");
printf("3 : Display Linked List\n");
printf("4 : Count Linked List\n");
printf("Others : Exit()\n");
printf("Enter your option:");
scanf("%d", &option);
switch (option) {
case 1:
insert();
break;
case 2:
delete();
break;
case 3:
display();
break;
case 4:
count();
break;
default:
```

```
break;
}
}
return 0;
}
void insert() {
printf("\nEnter Element for Insert Linked List : \n");
scanf("%d", &data);
temp_node = (DATA_NODE *) malloc(sizeof (DATA_NODE));
temp_node->value = data;
if (first_node == 0) {
first_node = temp_node;
} else {
head_node->next = temp_node;
}
temp_node->next = 0;
head_node = temp_node;
fflush(stdin);
}
void delete() {
int countvalue, pos, i = 0;
countvalue = count();
temp_node = first_node;
printf("\nDisplay Linked List : \n");
printf("\nEnter Position for Delete Element : \n");
scanf("%d", &pos);
if (pos > 0 && pos <= countvalue) {
if (pos == 1) {
temp_node = temp_node -> next;
first_node = temp_node;
printf("\nDeleted Successfully \n\n");
} else {
while (temp_node != 0) {
if (i == (pos - 1)) {
prev_node->next = temp_node->next;
if(i == (countvalue - 1))
{
head_node = prev_node;
}
printf("\nDeleted Successfully \n\n");
break;
} else {
i++;
prev_node = temp_node;
temp_node = temp_node -> next;
}
}
```

```
}
} else
printf("\nInvalid Position \n\n");
}
void display() {
int count = 0;
temp_node = first_node;
printf("\nDisplay Linked List : \n");
while (temp_node != 0) {
printf("# %d # ", temp_node->value);
count++;
temp_node = temp_node -> next;
}
printf("\nNo Of Items In Linked List : %d\n", count);
}
int count() {
int count = 0;
temp_node = first_node;
while (temp_node != 0) {
count++;
temp_node = temp_node -> next;
}
printf("\nNo Of Items In Linked List : %d\n", count);
return count;
}
```

OUTPUT

```
Singly Linked List Example - All Operations

Options
1 : Insert into Linked List
2 : Delete from Linked List
3 : Display Linked List
4 : Count Linked List
Others : Exit()
Enter your option:1

Enter Element for Insert Linked List :
33

Options
1 : Insert into Linked List
2 : Delete from Linked List
3 : Display Linked List
4 : Count Linked List
Others : Exit()
Enter your option:1

Enter Element for Insert Linked List :
55

Options
1 : Insert into Linked List
2 : Delete from Linked List
3 : Display Linked List
4 : Count Linked List
Others : Exit()
Enter your option:2

No Of Items In Linked List : 2

Display Linked List :

Enter Position for Delete Element :
2

Deleted Successfully


Options
1 : Insert into Linked List
2 : Delete from Linked List
3 : Display Linked List
4 : Count Linked List
Others : Exit()
Enter your option:3
```

```
Display Linked List :
# 33 #
No Of Items In Linked List : 1


Options
1 : Insert into Linked List
2 : Delete from Linked List
3 : Display Linked List
4 : Count Linked List
Others : Exit()
Enter your option:
```

DOUBLE LINKED LIST

SOURCE CODE

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
struct node *prev;
struct node *next;
int data;
};
struct node *head;
void insertion_beginning();
void insertion_last();
void insertion_specified();
void deletion_beginning();
void deletion_last();
void deletion_specified();
void display();
void search();
void main ()
{
int choice =0;
while(choice != 9)
{
printf("\n*********Main Menu*********\n");
printf("\nChoose one option from the following list ...\n");
printf("\n===============================================\n");
printf("\n1.Insert in begining\n2.Insert at last\n3.Insert at any random location\n4.Delete from Beginning\n5.Delete from
last\n6.Delete the node after the given data\n7.Search\n8.Show\n9.Exit\n");
printf("\nEnter your choice?\n");
scanf("\n%d",&choice);
switch(choice)
{
case 1:
insertion_beginning();
break;
case 2:
insertion_last();
break;
case 3:
insertion_specified();
break;
case 4:
deletion_beginning();
```

```
break;
case 5:
deletion_last();
break;
case 6:
deletion_specified();
break;
case 7:
search();
break;
case 8:
display();
break;
case 9:
exit(0);
break;
default:
printf("Please enter valid choice..");
}
}
}
void insertion_beginning()
{
struct node *ptr;
int item;
ptr = (struct node *)malloc(sizeof(struct node));
if(ptr == NULL)
{
printf("\nOVERFLOW");
}
else
{
printf("\nEnter Item value");
scanf("%d",&item);
if(head==NULL)
{
ptr->next = NULL;
ptr->prev=NULL;
ptr->data=item;
head=ptr;
}
else
{
ptr->data=item;
ptr->prev=NULL;
ptr->next = head;
head->prev=ptr;
```

```
head=ptr;
}
printf("\nNode inserted\n");
}
}
void insertion_last()
{
struct node *ptr,*temp;
int item;
ptr = (struct node *) malloc(sizeof(struct node));
if(ptr == NULL)
{
printf("\nOVERFLOW");
}
else
{
printf("\nEnter value");
scanf("%d",&item);
ptr->data=item;
if(head == NULL)
{
ptr->next = NULL;
ptr->prev = NULL;
head = ptr;
}
else
{
temp = head;
while(temp->next!=NULL)
{
temp = temp->next;
}
temp->next = ptr;
ptr ->prev=temp;
ptr->next = NULL;
}
}
printf("\nnode inserted\n");
}
void insertion_specified()
{
struct node *ptr,*temp;
int item,loc,i;
ptr = (struct node *)malloc(sizeof(struct node));
if(ptr == NULL)
{
printf("\n OVERFLOW");
```

```
}
else
{
temp=head;
printf("Enter the location");
scanf("%d",&loc);
for(i=0;i<loc;i++)
{
temp = temp->next;
if(temp == NULL)
{
printf("\n There are less than %d elements", loc);
return;
}
}
printf("Enter value");
scanf("%d",&item);
ptr->data = item;
ptr->next = temp->next;
ptr -> prev = temp;
temp->next = ptr;
temp->next->prev=ptr;
printf("\nnode inserted\n");
}
}
void deletion_beginning()
{
struct node *ptr;
if(head == NULL)
{
printf("\n UNDERFLOW");
}
else if(head->next == NULL)
{
head = NULL;
free(head);
printf("\nnode deleted\n");
}
else
{
ptr = head;
head = head -> next;
head -> prev = NULL;
free(ptr);
printf("\nnode deleted\n");
}
}
```

```
void deletion_last()
{
struct node *ptr;
if(head == NULL)
{
printf("\n UNDERFLOW");
}
else if(head->next == NULL)
{
head = NULL;
free(head);
printf("\nnode deleted\n");
}
else
{
ptr = head;
if(ptr->next != NULL)
{
ptr = ptr -> next;
}
ptr -> prev -> next = NULL;
free(ptr);
printf("\nnode deleted\n");
}
}
void deletion_specified()
{
struct node *ptr, *temp;
int val;
printf("\n Enter the data after which the node is to be deleted : ");
scanf("%d", &val);
ptr = head;
while(ptr -> data != val)
ptr = ptr -> next;
if(ptr -> next == NULL)
{
printf("\nCan't delete\n");
}
else if(ptr -> next -> next == NULL)
{
ptr ->next = NULL;
}
else
{
temp = ptr -> next;
ptr -> next = temp -> next;
temp -> next -> prev = ptr;
```

```
free(temp);
printf("\nnode deleted\n");
}}
void display()
{
struct node *ptr;
printf("\n printing values...\n");
ptr = head;
while(ptr != NULL)
{
printf("%d\n",ptr->data);
ptr=ptr->next;
}}
void search()
{
struct node *ptr;
int item,i=0,flag;
ptr = head;
if(ptr == NULL)
{
printf("\nEmpty List\n");
}
else
{
printf("\nEnter item which you want to search?\n");
scanf("%d",&item);
while (ptr!=NULL)
{
if(ptr->data == item)
{
printf("\nitem found at location %d ",i+1);
flag=0;
break;
}
else
{
flag=1;
}
i++;
ptr = ptr -> next;
}
if(flag==1)
{
printf("\nItem not found\n");
}
}
}
```

OUTPUT

```
*********Main Menu*********

Choose one option from the following list ...

===============================================

1.Insert in begining
2.Insert at last
3.Insert at any random location
4.Delete from Beginning
5.Delete from last
6.Delete the node after the given data
7.Search
8.Show
9.Exit

Enter your choice?
1

Enter Item value34

Node inserted

*********Main Menu*********

Choose one option from the following list ...

===============================================

1.Insert in begining
2.Insert at last
3.Insert at any random location
4.Delete from Beginning
5.Delete from last
6.Delete the node after the given data
7.Search
8.Show
9.Exit

Enter your choice?
2

Enter value45

node inserted

*********Main Menu*********
```

```
Choose one option from the following list ...

==============================================

1.Insert in begining
2.Insert at last
3.Insert at any random location
4.Delete from Beginning
5.Delete from last
6.Delete the node after the given data
7.Search
8.Show
9.Exit

Enter your choice?
3
Enter the location2

 There are less than 2 elements
*********Main Menu*********

Choose one option from the following list ...

==============================================

1.Insert in begining
2.Insert at last
3.Insert at any random location
4.Delete from Beginning
5.Delete from last
6.Delete the node after the given data
7.Search
8.Show
9.Exit

Enter your choice?
8

 printing values...
34
45

*********Main Menu*********

Choose one option from the following list ...

==============================================
```

```
Choose one option from the following list ...

========================================

1.Insert in begining
2.Insert at last
3.Insert at any random location
4.Delete from Beginning
5.Delete from last
6.Delete the node after the given data
7.Search
8.Show
9.Exit

Enter your choice?
9

Process returned 0 (0x0)   execution time : 41.043 s
Press any key to continue.
```

STACK

SOURCE CODE

```
#include <stdio.h>
#include<conio.h>
#include <stdlib.h>
#define SIZE 4
int top = -1, inp_array[SIZE];
void push();
void pop();
void show();
int main()
{
int choice;
while (1)
{
printf("\nPerform operations on the stack:");
printf("\n1.Push the element\n2.Pop the element\n3.Show\n4.End");
printf("\n\nEnter the choice: ");
scanf("%d", &choice);
switch (choice)
{
case 1:
push();
break;
case 2:
pop();
break;
case 3:
show();
break;
case 4:
exit(0);
default:
printf("\nInvalid choice!!");
}
}
}
void push()
{
int x;
if (top == SIZE - 1)
{
printf("\nOverflow!!");
}
```

```
else
{
printf("\nEnter the element to be added onto the stack: ");
scanf("%d", &x);
top = top + 1;
inp_array[top] = x;
}
}
void pop()
{
if (top == -1)
{
printf("\nUnderflow!!");
}
else
{
printf("\nPopped element: %d", inp_array[top]);
top = top - 1;
}
}
void show()
{
if (top == -1)
{
printf("\nUnderflow!!");
}
else
{
printf("\nElements present in the stack: \n");
for (int i = top; i >= 0; --i)
printf("%d\n", inp_array[i]);
}
}
```

OUTPUT

```
Perform operations on the stack:
1.Push the element
2.Pop the element
3.Show
4.End

Enter the choice: 1

Enter the element to be added onto the stack: 23

Perform operations on the stack:
1.Push the element
2.Pop the element
3.Show
4.End

Enter the choice: 1

Enter the element to be added onto the stack: 35

Perform operations on the stack:
1.Push the element
2.Pop the element
3.Show
4.End

Enter the choice: 2

Popped element: 35
Perform operations on the stack:
1.Push the element
2.Pop the element
3.Show
4.End

Enter the choice: 3

Elements present in the stack:
23

Perform operations on the stack:
1.Push the element
2.Pop the element
3.Show
4.End

Enter the choice: 4
```

EVALUATION OF ARITHMETIC OPERATION

SOURCE CODE

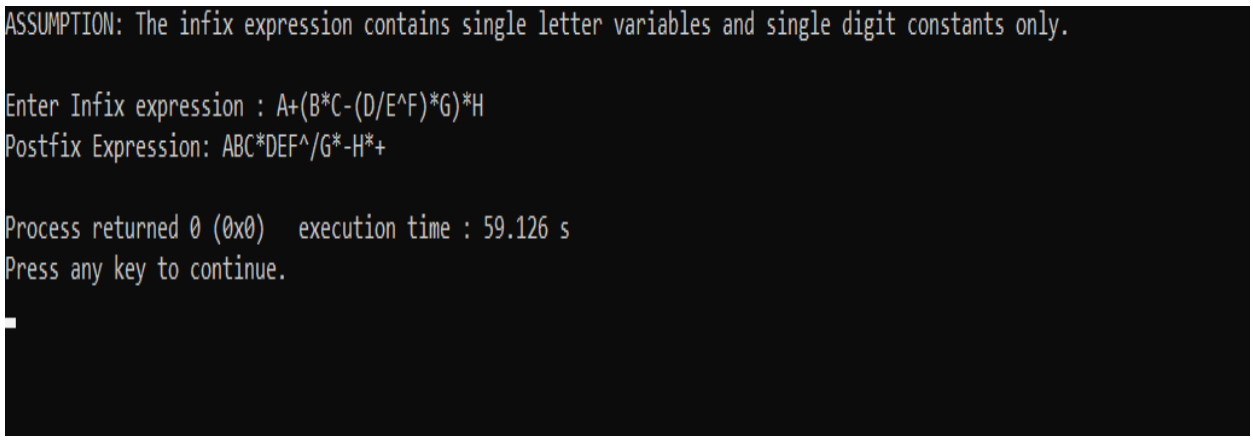
```
#include <stdio.h>
#include<conio.h>
#include <stdlib.h> /* for exit() */
#include <ctype.h> /* for isdigit(char ) */
#include <string.h>
#define SIZE 100
/* declared here as global variable because stack[]
* is used by more than one fucntions */
char stack[SIZE];
int top = -1;
clrscr();
/* define push operation */
void push(char item)
{
if (top >= SIZE - 1) {
printf("\nStack Overflow.");
}
else {
top = top + 1;
stack[top] = item;
}
}
/* define pop operation */
char pop()
{
char item;
if (top < 0) {
printf("stack under flow: invalid infix expression");
getchar();
/* underflow may occur for invalid expression */
/* where ( and ) are not matched */
exit(1);
}
else {
item = stack[top];
top = top - 1;
return (item);
}
}
int is_operator(char symbol)
{
```

```
if (symbol == '^' || symbol == '*' || symbol == '/' || symbol == '+' || symbol == '-') {
return 1;
}
else {
return 0;
}
}
int precedence(char symbol)
{
if (symbol == '^') /* exponent operator, highest precedence*/
{
return (3);
}
else if (symbol == '*' || symbol == '/') {
return (2);
}
else if (symbol == '+' || symbol == '-') /* lowest precedence */
{
return (1);
}
else {
return (0);
}
}
void InfixToPostfix(char infix_exp[], char postfix_exp[])
{
int i, j;
char item;
char x;
push('('); /* push '(' onto stack */
strcat(infix_exp, ")"); /* add ')' to infix expression */
i = 0;
j = 0;
item = infix_exp[i]; /* initialize before loop*/
while (item != '\0') /* run loop till end of infix expression */
{
if (item == '(') {
push(item);
}
else if (isdigit(item) || isalpha(item)) {
postfix_exp[j] = item; /* add operand symbol to postfix expr */
j++;
}
else if (is_operator(item) == 1) /* means symbol is operator */
{
x = pop();
while (is_operator(x) == 1 && precedence(x) >= precedence(item)) {
```

```
postfix_exp[j] = x; /* so pop all higher precendence operator and */
j++;
x = pop(); /* add them to postfix expresion */
}
push(x);
push(item); /* push current oprerator symbol onto stack */
}
else if (item == ')') /* if current symbol is ')' then */
{
x = pop(); /* pop and keep popping until */
while (x != '(') /* '(' encounterd */
{
postfix_exp[j] = x;
j++;
x = pop();
}
}
else { /* if current symbol is neither operand not '(' nor ')' and nor
operator */
printf("\nInvalid infix Expression.\n"); /* the it is illegeal  symbol */
getchar();
exit(1);
}
i++;
item = infix_exp[i]; /* go to next symbol of infix expression */
} /* while loop ends here */
if (top > 0) {
printf("\nInvalid infix Expression.\n"); /* the it is illegeal  symbol */
getchar();
exit(1);
}
if (top > 0) {
printf("\nInvalid infix Expression.\n"); /* the it is illegeal  symbol */
getchar();
exit(1);
}
postfix_exp[j] = '\0'; /* add sentinel else puts() fucntion */
/* will print entire postfix[] array upto SIZE */
}
/* main function begins */
int main()
{
char infix[SIZE], postfix[SIZE]; /* declare infix string and postfix string */
printf("ASSUMPTION: The infix expression contains single letter variables and single digit constants only.\n");
printf("\nEnter Infix expression : ");
gets(infix);
```

```
InfixToPostfix(infix, postfix); /* call to convert */
printf("Postfix Expression: ");
puts(postfix); /* print postfix expression */
return 0;
}
```

OUTPUT


ASSUMPTION: The infix expression contains single letter variables and single digit constants only.
Enter Infix expression : A+(B*C-(D/E^F)*G)*H
Postfix Expression: ABC*DEF^/G*-H*+
Process returned 0 (0x0) execution time : 59.126 s
Press any key to continue.

BINARY TREE TRAVERSAL

SOURCE CODE

```
#include <stdio.h>
#include<conio.h>
#include <stdlib.h>
struct tnode {
int data;
struct tnode *left, *right;
};
struct tnode *root = NULL;
/* creating node of the tree  and fill the given data */
struct tnode * createNode(int data) {
struct tnode *newNode;
newNode  = (struct tnode *) malloc(sizeof(struct tnode));
newNode->data = data;
newNode->left = NULL;
newNode->right = NULL;
return (newNode);
}
/* inserting a new node into the tree */
void insertion(struct tnode **node, int data) {
if (!*node) {
*node = createNode(data);
} else if (data < (*node)->data) {
insertion(&(*node)->left, data);
} else if (data > (*node)->data) {
insertion(&(*node)->right, data);
}}
/* post order tree traversal */
void postOrder(struct tnode *node) {
if (node) {
postOrder(node->left);
postOrder(node->right);
printf("%d  ", node->data);
}
return;
}
/* pre order tree traversal */
void preOrder(struct tnode *node) {
if (node) {
printf("%d  ", node->data);
preOrder(node->left);
preOrder(node->right);
}
```

```
return;
}
/* inorder tree traversal */
void inOrder(struct tnode *node) {
if (node) {
inOrder(node->left);
printf("%d  ", node->data);
inOrder(node->right);
}
return;
}
int main() {
int data, ch;
while (1) {
printf("\n1. Insertion\n2. Pre-order\n");
printf("3. Post-order\n4. In-order\n");
printf("5. Exit\nEnter your choice:");
scanf("%d", &ch);
switch (ch) {
case 1:
printf("Enter ur data:");
scanf("%d", &data);
insertion(&root, data);
break;
case 2:
preOrder(root);
break;
case 3:
postOrder(root);
break;
case 4:
inOrder(root);
break;
case 5:
exit(0);
default:
printf("U've entered wrong opetion\n");
break;
}
}
return 0;
}
```

OUTPUT

```
1. Insertion
2. Pre-order
3. Post-order
4. In-order
5. Exit
Enter your choice:1
Enter ur data:23

1. Insertion
2. Pre-order
3. Post-order
4. In-order
5. Exit
Enter your choice:1
Enter ur data:56

1. Insertion
2. Pre-order
3. Post-order
4. In-order
5. Exit
Enter your choice:1
Enter ur data:89

1. Insertion
2. Pre-order
3. Post-order
4. In-order
5. Exit
Enter your choice:1
Enter ur data:12

1. Insertion
2. Pre-order
3. Post-order
4. In-order
5. Exit
Enter your choice:2
23  12  56  89
1. Insertion
2. Pre-order
3. Post-order
4. In-order
5. Exit
Enter your choice:3
12  89  56  23
```

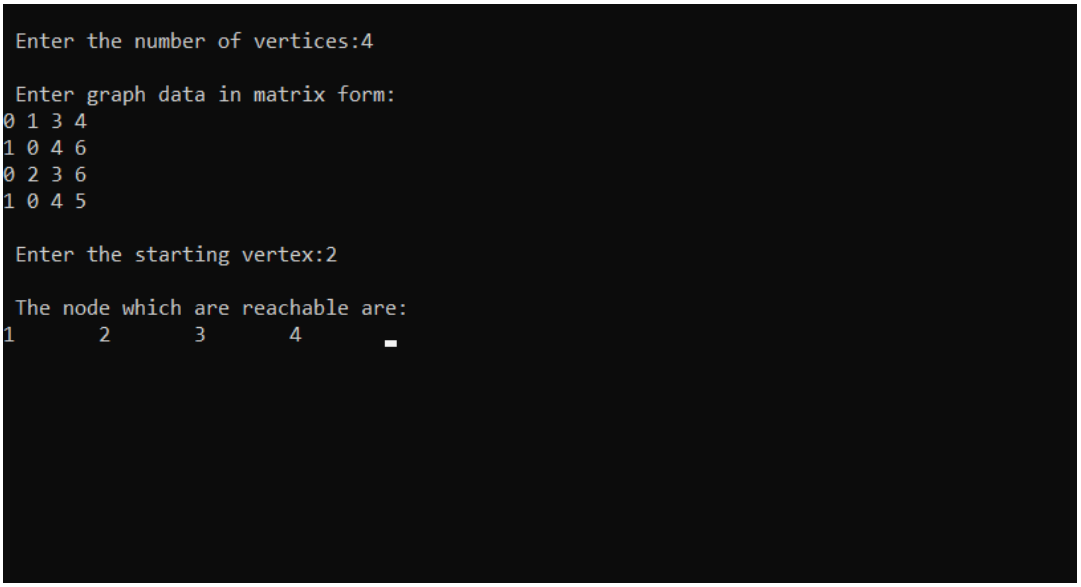
```
1. Insertion
2. Pre-order
3. Post-order
4. In-order
5. Exit
Enter your choice:5

Process returned 0 (0x0)   execution time : 77.233 s
Press any key to continue.
```

BREADTH FIRST SEARCH

SOURCE CODE

```
#include<stdio.h>
#include<conio.h>
int a[20][20],q[20],visited[20],n,I,j,f=0,r=-1;
clrscr();
void bfs(int v) {
for (i=1;i<=n;i++)
if(a[v][i] && !visited[i])
q[++r]=I;
if(f<=r) {
visited[q[f]]=1;
bfs(q[f++]);
}
}
void main() {
int v;
printf("\n Enter the number of vertices:");
scanf("%d",&n);
for (i=1;i<=n;i++) {
q[i]=0;
visited[i]=0;
}
printf("\n Enter graph data in matrix form:\n");
for (i=1;i<=n;i++)
for (j=1;j<=n;j++)
scanf("%d",&a[i][j]);
printf("\n Enter the starting vertex:");
scanf("%d",&v);
bfs(v);
printf("\n The node which are reachable are:\n");
for (i=1;i<=n;i++)
if(visited[i])
printf("%d\t",i); else
printf("\n Bfs is not possible");
getch();
}
```

OUTPUT


Enter the number of vertices:4
Enter graph data in matrix form:
0 1 3 4
1 0 4 6
0 2 3 6
1 0 4 5
Enter the starting vertex:2
The node which are reachable are:
1 2 3 4

DEPTH FIRST SEARCH

SOURCE CODE

```
#include<stdio.h>
#include<conio.h>
int a[20][20],reach[20],n;
void dfs(int v) {
int i;
reach[v]=1;
for (i=1;i<=n;i++)
if(a[v][i] && !reach[i]) {
printf("\n %d->%d",v,i);
dfs(i);
}
}
void main() {
int i,j,count=0;
clrscr();
printf("\n Enter number of vertices:");
scanf("%d",&n);
for (i=1;i<=n;i++) {
reach[i]=0;
for (j=1;j<=n;j++)
a[i][j]=0;
}
printf("\n Enter the adjacency matrix:\n");
for (i=1;i<=n;i++)
for (j=1;j<=n;j++)
scanf("%d",&a[i][j]);
dfs(1);
printf("\n");
for (i=1;i<=n;i++) {
if(reach[i])
count++;
}
if(count==n)
printf("\n Graph is connected"); else
printf("\n Graph is not connected");
getch();
}
```

OUTPUT


Enter number of vertices:3
Enter the adjacency matrix:
1 0 2
2 6 9
0 5 1
1->3
3->2
Graph is connected

PRIMS ALGORITHM

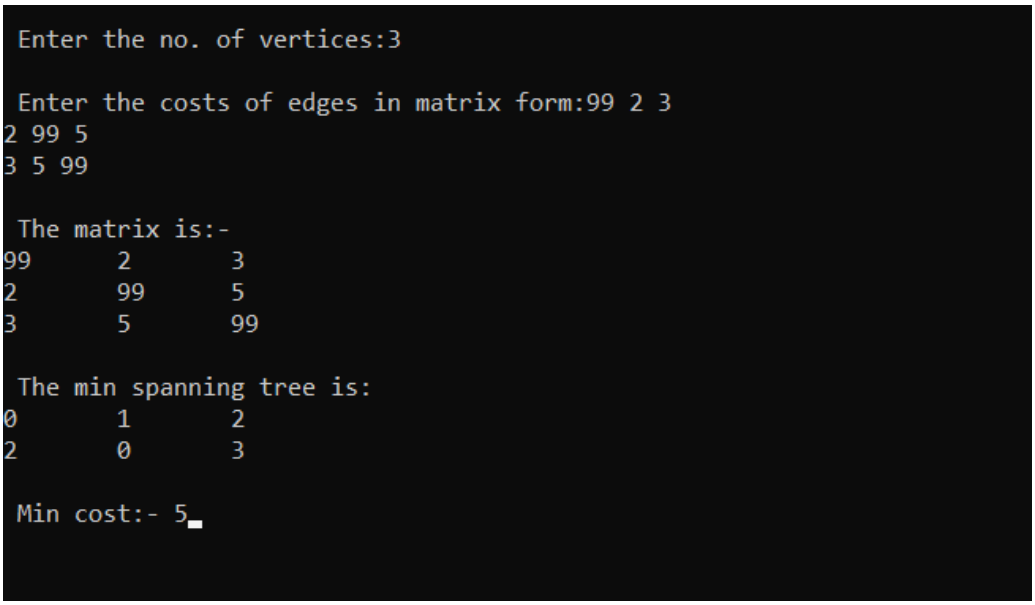
SOURCE CODE

```
#include<stdio.h>
#include<conio.h>
int n, cost[10][10];
void prim()
{
int i,j,k,l,x,nr[10],temp,min_cost=0,tree[10][3];
/* For first smallest edge */
clrscr();
temp=cost[0][0];
for(i=0;i < n;i++)
{
for(j=0;j < n;j++)
{
if(temp > cost[i][j])
{
temp=cost[i][j];
k=i;
l=j;
}
}
}
/* Now we have fist smallest edge in graph */
tree[0][0]=k;
tree[0][1]=l;
tree[0][2]=temp;
min_cost=temp;
/*        Now we have to find min dis of each
vertex from either k or l
by initialising nr[] array */
for(i=0;i< n;i++)
{
if(cost[i][k]< cost[i][l])
nr[i]=k;
else
nr[i]=l;
}
/* To indicate visited vertex initialise nr[] for them to 100 */
nr[k]=100;
nr[l]=100;
/* Now find out remaining n-2 edges */
temp=99;
for(i=1;i< n-1;i++)
```

```
{
for(j=0;j< n;j++)
{
if(nr[j]!=100 && cost[j][nr[j]] < temp)
{
temp=cost[j][nr[j]];
x=j;
}
}
/* Now i have got next vertex */
tree[i][0]=x;
tree[i][1]=nr[x];
tree[i][2]=cost[x][nr[x]];
min_cost=min_cost+cost[x][nr[x]];
nr[x]=100;
/* Now find if x is nearest to any vertex
than its previous near value */
for(j=0;j< n;j++)
{
if(nr[j]!=100 && cost[j][nr[j]] > cost[j][x])
nr[j]=x;
}
temp=99;
}
/* Now i have the answer, just going to print it */
printf("\n The min spanning tree is: \n");
for(i=0;i < n-1;i++)
{
for(j=0;j < 3;j++)
printf("%d\t", tree[i][j]);
printf("\n");
}
printf("\n Min cost:- %d", min_cost);
}
void main()
{
int i,j;
printf("\n Enter the no. of vertices:");
scanf("%d", &n);
printf ("\n Enter the costs of edges in matrix form:");
for(i=0;i< n;i++)
for(j=0;j< n;j++)
scanf("%d",&cost[i][j]);
printf("\n The matrix is: \n");
```

```
for(i=0;i< n;i++)
{
for(j=0;j < n;j++)
printf("%d\t",cost[i][j]);
printf("\n");
}
prim();
getch();
}
```

OUTPUT


Enter the no. of vertices:3
Enter the costs of edges in matrix form:99 2 3
2 99 5
3 5 99
The matrix is:-
99 2 3
2 99 5
3 5 99
The min spanning tree is:
0 1 2
2 0 3
Min cost:- 5

DIJKSTRA'S ALGORITHM

SOURCE CODE

```
#include<stdio.h>
#include<conio.h>
#define INFINITY 9999
#define MAX 10
void dijkstra(int G[MAX][MAX],int n,int startnode);
int main()
{
int G[MAX][MAX],i,j,n,u;
clrscr();
printf("Enter no. of vertices:");
scanf("%d",&n);
printf("\nEnter the adjacency matrix:\n");
for(i=0;i<n;i++)
for(j=0;j<n;j++)
scanf("%d",&G[i][j]);
printf("\nEnter the starting node:");
scanf("%d",&u);
dijkstra(G,n,u);
return 0;
}
void dijkstra(int G[MAX][MAX],int n,int startnode)
{
int cost[MAX][MAX],distance[MAX],pred[MAX];
int visited[MAX],count,mindistance,nextnode,i,j;
//pred[] stores the predecessor of each node
//count gives the number of nodes seen so far
//create the cost matrix
for(i=0;i<n;i++)
for(j=0;j<n;j++)
if(G[i][j]==0)
cost[i][j]=INFINITY;
else
cost[i][j]=G[i][j];
//initialize pred[],distance[] and visited[]
for(i=0;i<n;i++)
{
distance[i]=cost[startnode][i];
pred[i]=startnode;
visited[i]=0;
}
distance[startnode]=0;
visited[startnode]=1;
```

```
count=1;
while(count<n-1)
{
mindistance=INFINITY;
//nextnode gives the node at minimum distance
for(i=0;i<n;i++)
if(distance[i]<mindistance&&!visited[i])
{
mindistance=distance[i];
nextnode=i;
}
//check if a better path exists through nextnode
visited[nextnode]=1;
for(i=0;i<n;i++)
if(!visited[i])
if(mindistance+cost[nextnode][i]<distance[i])
{
distance[i]=mindistance+cost[nextnode][i];
pred[i]=nextnode;
}
count++;
}
//print the path and distance of each node
for(i=0;i<n;i++)
if(i!=startnode)
{
printf("\nDistance of node%d=%d",i,distance[i]);
printf("\nPath=%d",i);
j=i;
do
{
j=pred[j];
printf("<-%d",j);
}while(j!=startnode);
}
}
```

OUTPUT

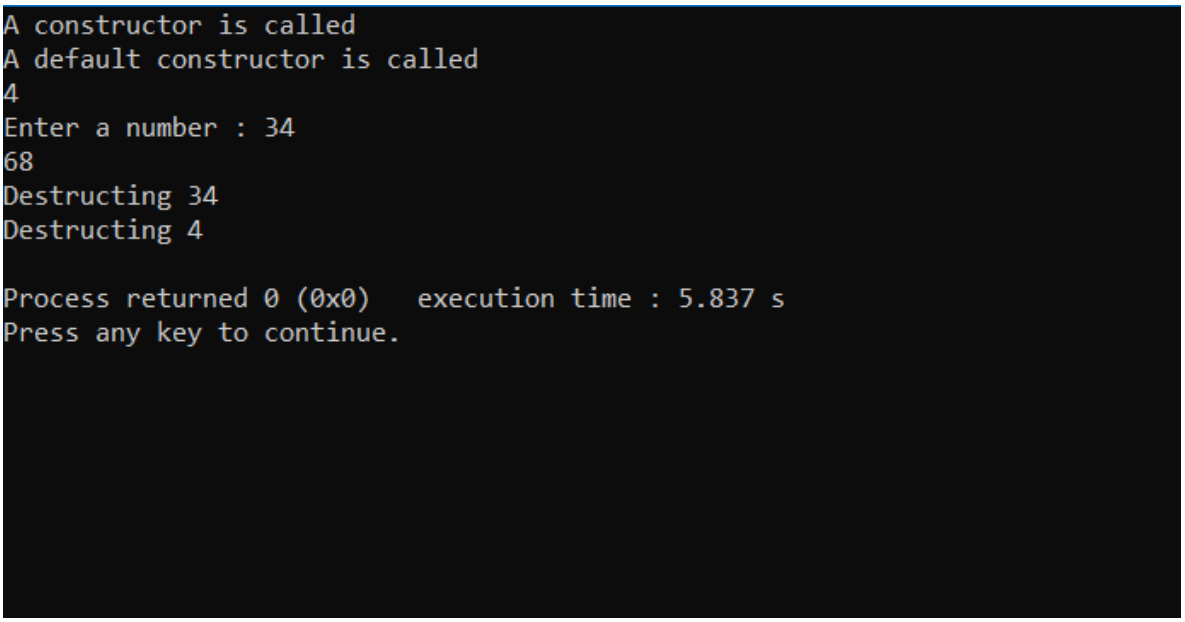
```
Enter no. of vertices:5

Enter the adjacency matrix:
0 10 0 30 100
10 0 50 0 0
0 50 0 20 10
30 0 20 0 60
100 0 10 60 0

Enter the starting node:0

Distance of node1=10
Path=1<-0
Distance of node2=50
Path=2<-3<-0
Distance of node3=30
Path=3<-0
Distance of node4=60
Path=4<-2<-3<-0
Process returned 0 (0x0)   execution time : 62.362 s
Press any key to continue.
```

IMPLEMENTATION OF CLASSES AND OBJECTS WITH CONSTRUCTORS AND DESTRUCTORS

SOURCE CODE

```
#include<iostream>
using namespace std;
class CAdd
{
public:
int one;
CAdd(int two)
{
cout << "A constructor is called" << endl;
one=two;
}
CAdd()
{
cout << "A default constructor is called " << endl;
}
~CAdd()
{
cout << "Destructing " << one << endl;
}
int add()
{
return(one+one);
}
};
int main()
{
CAdd myobj1(4);
CAdd myobj2;
cout << myobj1.one << endl;
cout << "Enter a number : " ;
cin >> myobj2.one;
cout << myobj2.add() << endl;
return(0);
}
```

OUTPUT

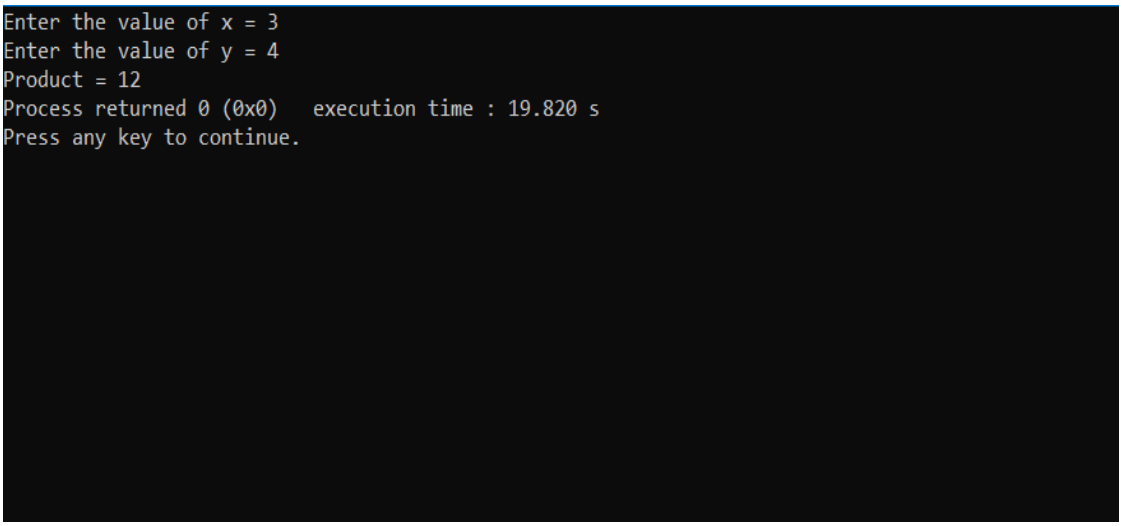

A constructor is called
A default constructor is called
4
Enter a number : 34
68
Destructing 34
Destructing 4
Process returned 0 (0x0)   execution time : 5.837 s
Press any key to continue.

SINGLE INHERITANCE

SOURCE CODE

```
#include <iostream>
using namespace std;
class base    //single base class
{
public:
int x;
void getdata()
{
cout << "Enter the value of x = "; cin >> x;
}
};
class derive : public base    //single derived class
{
private:
int y;
public:
void readdata()
{
cout << "Enter the value of y = "; cin >> y;
}
void product()
{
cout << "Product = " << x * y;
}
};
int main()
{
derive a;     //object of derived class
a.getdata();
a.readdata();
a.product();
return 0;
}         //end of program
```

OUTPUT

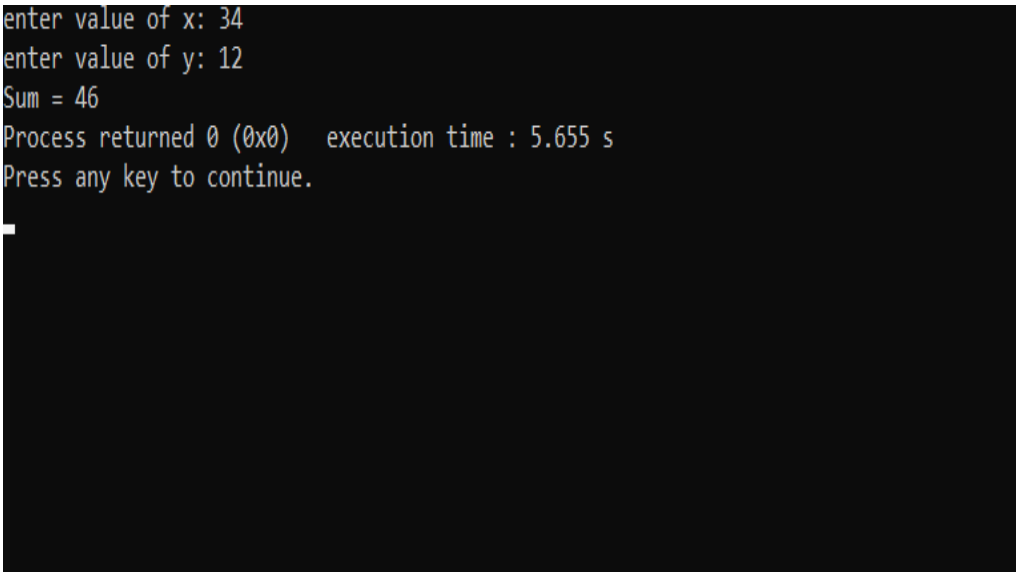

Enter the value of x = 3
Enter the value of y = 4
Product = 12
Process returned 0 (0x0)   execution time : 19.820 s
Press any key to continue.

MULTIPLE INHERITANCE

SOURCE CODE

```
#include<iostream>
using namespace std;
class A
{
public:
int x;
void getx()
{
cout << "enter value of x: "; cin >> x;
}
};
class B
{
public:
int y;
void gety()
{
cout << "enter value of y: "; cin >> y;
}
};
class C : public A, public B   //C is derived from class A and class B
{
public:
void sum()
{
cout << "Sum = " << x + y;
}
};
int main()
{
C obj1; //object of derived class C
obj1.getx();
obj1.gety();
obj1.sum();
return 0;
}          //end of program
```

OUTPUT

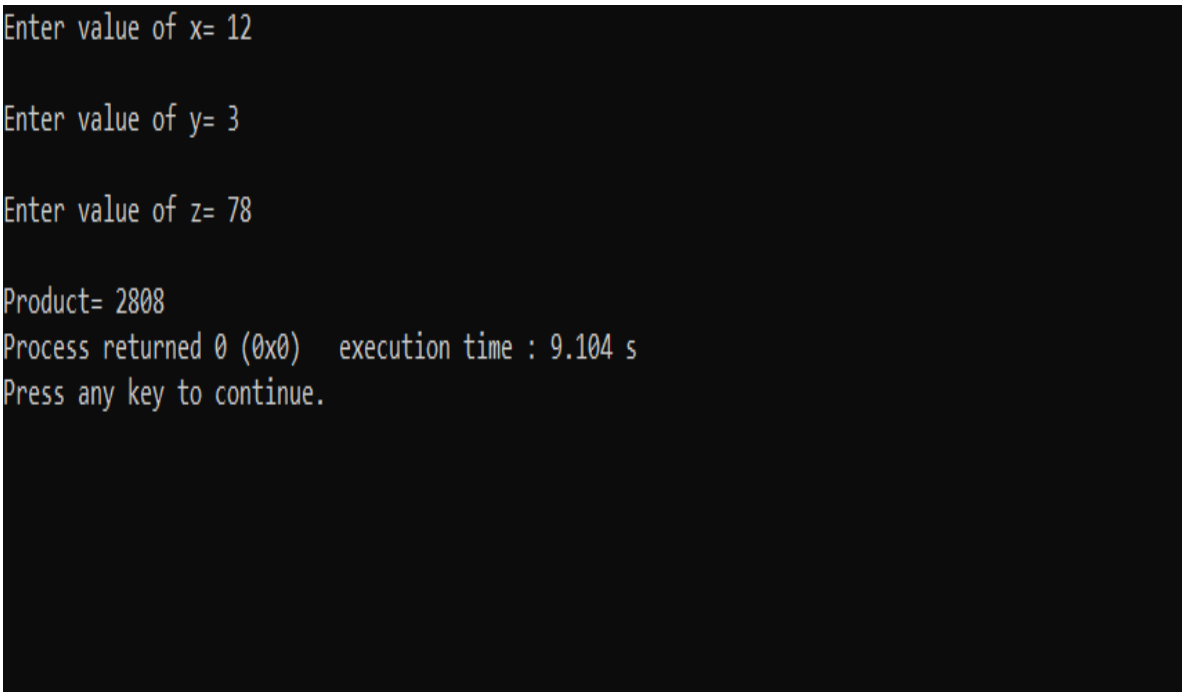

enter value of x: 34
enter value of y: 12
Sum = 46
Process returned 0 (0x0)   execution time : 5.655 s
Press any key to continue.

MULTILEVEL INHERITANCE

SOURCE CODE

```
#include <iostream>
using namespace std;
class base //single base class
{
public:
int x;
void getdata()
{
cout << "Enter value of x= "; cin >> x;
}};
class derive1 : public base // derived class from base class
{
public:
int y;
void readdata()
{
cout << "\nEnter value of y= "; cin >> y;
}
};
class derive2 : public derive1   // derived from class derive1
{
private:
int z;
public:
void indata()
{
cout << "\nEnter value of z= "; cin >> z;
}
void product()
{
cout << "\nProduct= " << x * y * z;
}};
int main()
{
derive2 a;      //object of derived class
a.getdata();
a.readdata();
a.indata();
a.product();
return 0;
}
```

OUTPUT


Enter value of x= 12
Enter value of y= 3
Enter value of z= 78
Product= 2808
Process returned 0 (0x0)   execution time : 9.104 s
Press any key to continue.

HYBRID INHERITANCE

SOURCE CODE

```
#include<iostream>
using namespace std;
class Person
{
int id;
char name[200];
public:
void accept_person_details()
{
cout<<"\n ---------------------------------------------- \n";
cout<<"\n Enter Id              :  ";
cin>>id;
cout<<"\n Enter Name            :  ";
cin>>name;
}
void display_person_details()
{
cout<<"\n ---------------------------------------------- \n";
cout<<"\n Id                    :  "<<id;
cout<<"\n Name                  :  "<<name;
}
};
class Teaching : public Person
{
char subject_name[100];
char teacher_name[200];
public:
void accept_teacher_details()
{
accept_person_details();
cout<<"\n Enter Subject Name     :  ";
cin>>subject_name;
cout<<"\n Enter Teacher Name     :  ";
cin>>teacher_name;
}
void display_teacher_details()
{
display_person_details();
cout<<"\n Subject Name           :  "<<subject_name;
cout<<"\n Teacher Name           :  "<<teacher_name;
}
};
class NonTeaching : public Person
{
```

```
char dept_name[200];
public:
void accept_nonteaching_details()
{
cout<<"\n Enter Department Name  :  ";
cin>>dept_name;
}
void display_nonteaching_details()
{
cout<<"\n Department Name        :  "<<dept_name;
}
};
class Instructor : public NonTeaching, public Teaching
{
public:
void accept_instructor_details()
{
accept_teacher_details();
accept_nonteaching_details();
}
void display_instructor_details()
{
display_teacher_details();
display_nonteaching_details();
}
};
int main()
{
Instructor *inst;
int cnt, i;
cout<<"\n Enter No. of Instructor Details You Want?  :  ";
cin>>cnt;
inst=new Instructor[cnt];
for(i=0; i<cnt; i++)
{
inst[i].accept_instructor_details();
}
for(i=0; i<cnt; i++)
{
inst[i].display_instructor_details();
}
return 0;
}
```

OUTPUT

```
Enter No. of Instructor Details You Want?  :  2

-------------------------------------------------

Enter Id               :  1

Enter Name             :  XXXXX

Enter Subject Name     :  Account

Enter Teacher Name     :  yyyyy

Enter Department Name  :  commerce

-------------------------------------------------

Enter Id               :  2

Enter Name             :  xxxxx

Enter Subject Name     :  biology

Enter Teacher Name     :  yyyyy

Enter Department Name  :  science

-------------------------------------------------

Id                     :  1
Name                   :  XXXXX
Subject Name           :  Account
Teacher Name           :  yyyyy
Department Name        :  commerce
-------------------------------------------------

Id                     :  2
Name                   :  xxxxx
Subject Name           :  biology
Teacher Name           :  yyyyy
Department Name        :  science
Process returned 0 (0x0)   execution time : 74.792 s
Press any key to continue.
```

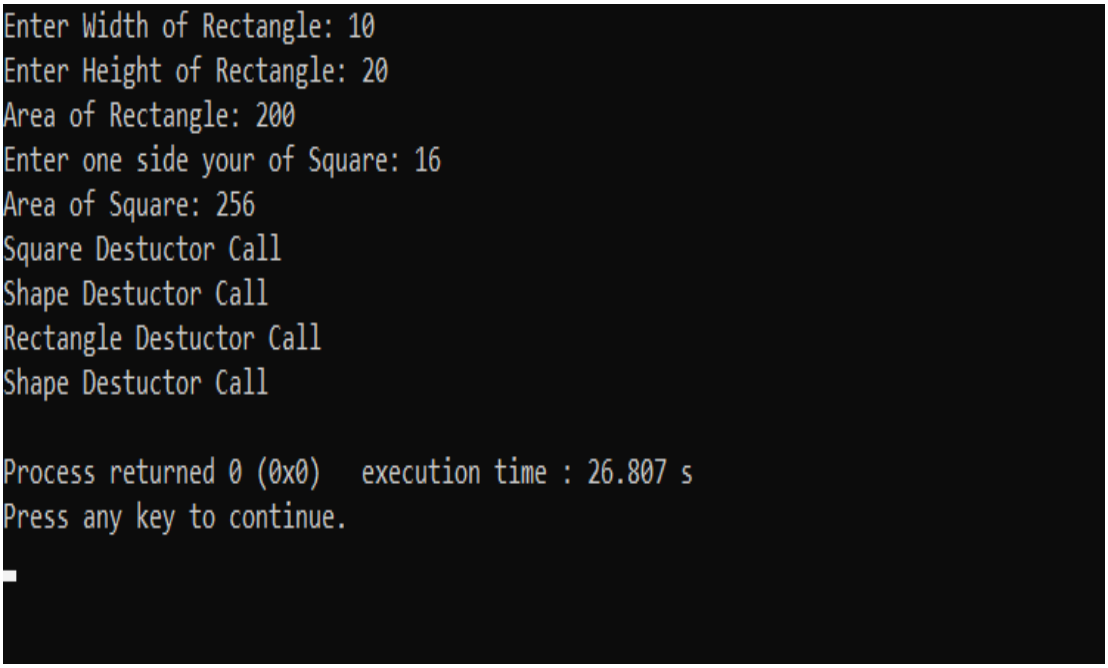
IMPLEMENTATION OF VIRTUAL FUNCTIONS TO DEMONSTRATE THE USE OF RUN TIME POLYMORPHISM

SOURCE CODE

```
#include <fstream>
#include <iostream>
using namespace std;
// Declaration of Base class
class Shape {
public:
// Usage of virtual constructor
virtual void calculate()
{
cout << "Area of your Shape ";
}
// usage of virtual Destuctor to avoid memory leak
virtual ~Shape()
{
cout << "Shape Destuctor Call\n";
}
};
// Declaration of Derived class
class Rectangle : public Shape {
public:
int width, height, area;
void calculate()
{
cout << "Enter Width of Rectangle: ";
cin >> width;
cout << "Enter Height of Rectangle: ";
cin >> height;
area = height * width;
cout << "Area of Rectangle: " << area << "\n";
}
// Virtual Destuctor for every Derived class
virtual ~Rectangle()
{
cout << "Rectangle Destuctor Call\n";
}
};
// Declaration of 2nd derived class
class Square : public Shape {
public:
int side, area;
```

```
void calculate()
{
cout << "Enter one side your of Square: ";
cin >> side;
area = side * side;
cout << "Area of Square: " << area << "\n";
}
// Virtual Destuctor for every Derived class
virtual ~Square()
{
cout << "Square Destuctor Call\n";
}
};
int main()
{
// base class pointer
Shape* S;
Rectangle r;
// initialization of reference variable
S = &r;
// calling of Rectangle function
S->calculate();
Square sq;
// initialization of reference variable
S = &sq;
// calling of Square function
S->calculate();
// return 0 to tell the program executed
// successfully
return 0;
}
```

OUTPUT


Enter Width of Rectangle: 10
Enter Height of Rectangle: 20
Area of Rectangle: 200
Enter one side your of Square: 16
Area of Square: 256
Square Destuctor Call
Shape Destuctor Call
Rectangle Destuctor Call
Shape Destuctor Call
Process returned 0 (0x0) execution time : 26.807 s
Press any key to continue.

IMPLEMENTATION OF QUEUE

SOURCE CODE

```
#include <iostream>
using namespace std;
int queue[100], n = 100, front = - 1, rear = - 1;
void Insert() {
int val;
if (rear == n - 1)
cout<<"Queue Overflow"<<endl;
else {
if (front == - 1)
front = 0;
cout<<"Insert the element in queue : "<<endl;
cin>>val;
rear++;
queue[rear] = val;
}
}
void Delete() {
if (front == - 1 || front > rear) {
cout<<"Queue Underflow ";
return ;
} else {
cout<<"Element deleted from queue is : "<< queue[front] <<endl;
front++;;
}
}
void Display() {
if (front == - 1)
cout<<"Queue is empty"<<endl;
else {
cout<<"Queue elements are : ";
for (int i = front; i <= rear; i++)
cout<<queue[i]<<" ";
cout<<endl;
}
}
int main() {
int ch;
cout<<"1) Insert element to queue"<<endl;
cout<<"2) Delete element from queue"<<endl;
cout<<"3) Display all the elements of queue"<<endl;
cout<<"4) Exit"<<endl;
```

```
do {
cout<<"Enter your choice : "<<endl;
cin>>ch;
switch (ch) {
case 1: Insert();
break;
case 2: Delete();
break;
case 3: Display();
break;
case 4: cout<<"Exit"<<endl;
break;
default: cout<<"Invalid choice"<<endl;
}
} while(ch!=4);
return 0;
}
```

OUTPUT

```
1) Insert element to queue
2) Delete element from queue
3) Display all the elements of queue
4) Exit
Enter your choice :
1
Insert the element in queue :
4
Enter your choice :
1
Insert the element in queue :
3
Enter your choice :
1
Insert the element in queue :
5
Enter your choice :
2
Element deleted from queue is : 4
Enter your choice :
3
Queue elements are : 3 5
Enter your choice :
4
Exit

Process returned 0 (0x0)   execution time : 25.079 s
Press any key to continue.
```