Data Structure Practicals

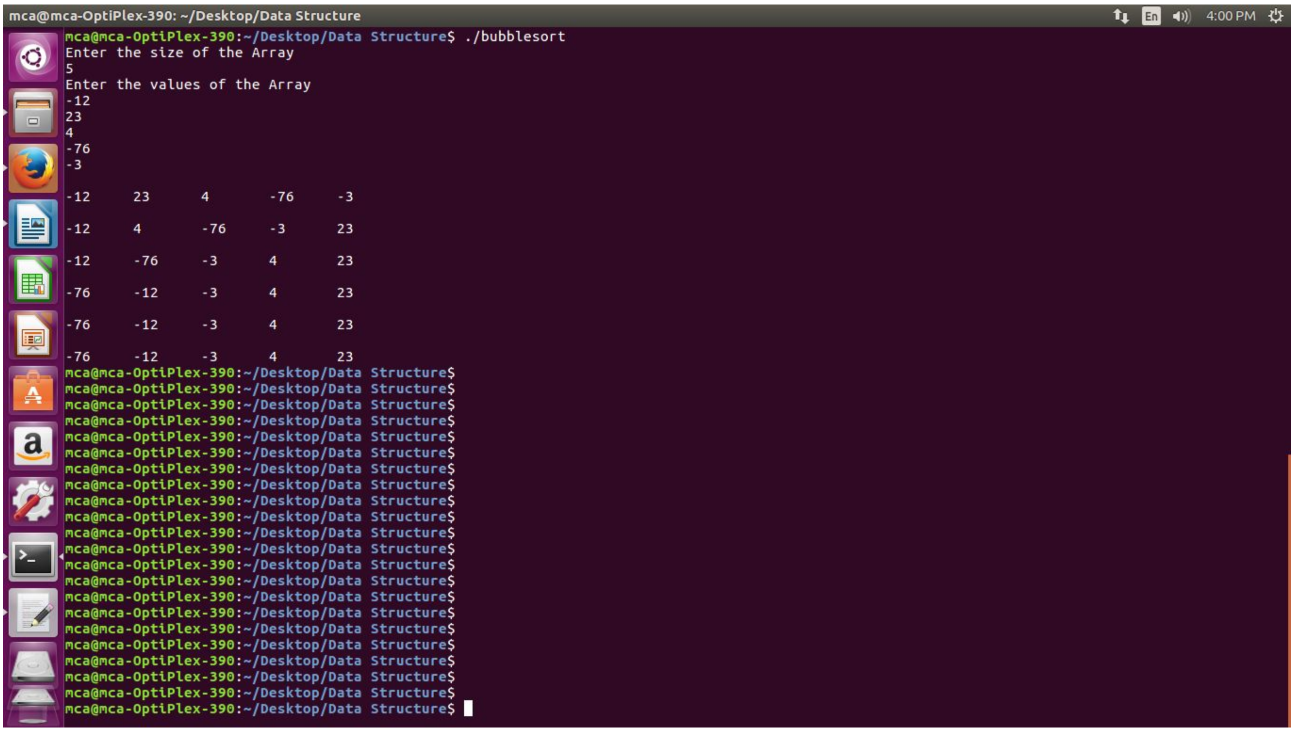
1. **Bubble Sort:-**

Code:-

```
#include<iostream>
#define max 25
using namespace std;


class bubble
{

	int n, A[max],temp,count;

	public:
		bubble()
		{
			n = 0;
			count = 0;
			temp = 0;
		}

		void getdata()
		{
			cout<<"Enter the size of the Array"<<endl;
			cin>>n;
			A[n];
			cout<<"Enter the values of the Array"<<endl;
			for(int i=0; i<n; i++)
			{
				cin>>A[i];
			}
		}

		void display()
		{
			cout<<endl;
			for(int i=0; i<n; i++)
			{
				cout<<A[i]<<"\t";
			}
			cout<<endl;
		}
```

```
        void sort()
        {
                for(int i=0; i<n-1; i++)
                {
                        count = 0;
                        for(int j=0; j<n-i-1; j++)
                        {
                                if(A[j]>A[j+1])
                                {
                                        temp = A[j];
                                        A[j] = A[j+1];
                                        A[j+1] = temp;
                                        count++;
                                }
                        }

                        cout<<"Pass = ";
                        display();
                        if(count == 0)
                        {
                                break;
                        }
                }
        }
};

int main()
{

bubble obj;

obj.getdata();
obj.display();
obj.sort();
cout<<"After sort :- ";obj.display();

return 0;
}
```

Screenshots:-



```
mca@mca-OptiPlex-390:~/Desktop/Data Structure$ ./bubblesort
Enter the size of the Array
5
Enter the values of the Array
-12
23
4
-76
-3

-12     23      4       -76     -3

-12     4       -76     -3      23

-12     -76     -3      4       23

-76     -12     -3      4       23

-76     -12     -3      4       23

-76     -12     -3      4       23
mca@mca-OptiPlex-390:~/Desktop/Data Structure$
mca@mca-OptiPlex-390:~/Desktop/Data Structure$
mca@mca-OptiPlex-390:~/Desktop/Data Structure$
mca@mca-OptiPlex-390:~/Desktop/Data Structure$
mca@mca-OptiPlex-390:~/Desktop/Data Structure$
mca@mca-OptiPlex-390:~/Desktop/Data Structure$
mca@mca-OptiPlex-390:~/Desktop/Data Structure$
mca@mca-OptiPlex-390:~/Desktop/Data Structure$
mca@mca-OptiPlex-390:~/Desktop/Data Structure$
mca@mca-OptiPlex-390:~/Desktop/Data Structure$
mca@mca-OptiPlex-390:~/Desktop/Data Structure$
mca@mca-OptiPlex-390:~/Desktop/Data Structure$
mca@mca-OptiPlex-390:~/Desktop/Data Structure$
mca@mca-OptiPlex-390:~/Desktop/Data Structure$
mca@mca-OptiPlex-390:~/Desktop/Data Structure$
mca@mca-OptiPlex-390:~/Desktop/Data Structure$
mca@mca-OptiPlex-390:~/Desktop/Data Structure$
mca@mca-OptiPlex-390:~/Desktop/Data Structure$
mca@mca-OptiPlex-390:~/Desktop/Data Structure$
mca@mca-OptiPlex-390:~/Desktop/Data Structure$
mca@mca-OptiPlex-390:~/Desktop/Data Structure$
mca@mca-OptiPlex-390:~/Desktop/Data Structure$ 
```

2. **Quick Sort:-**

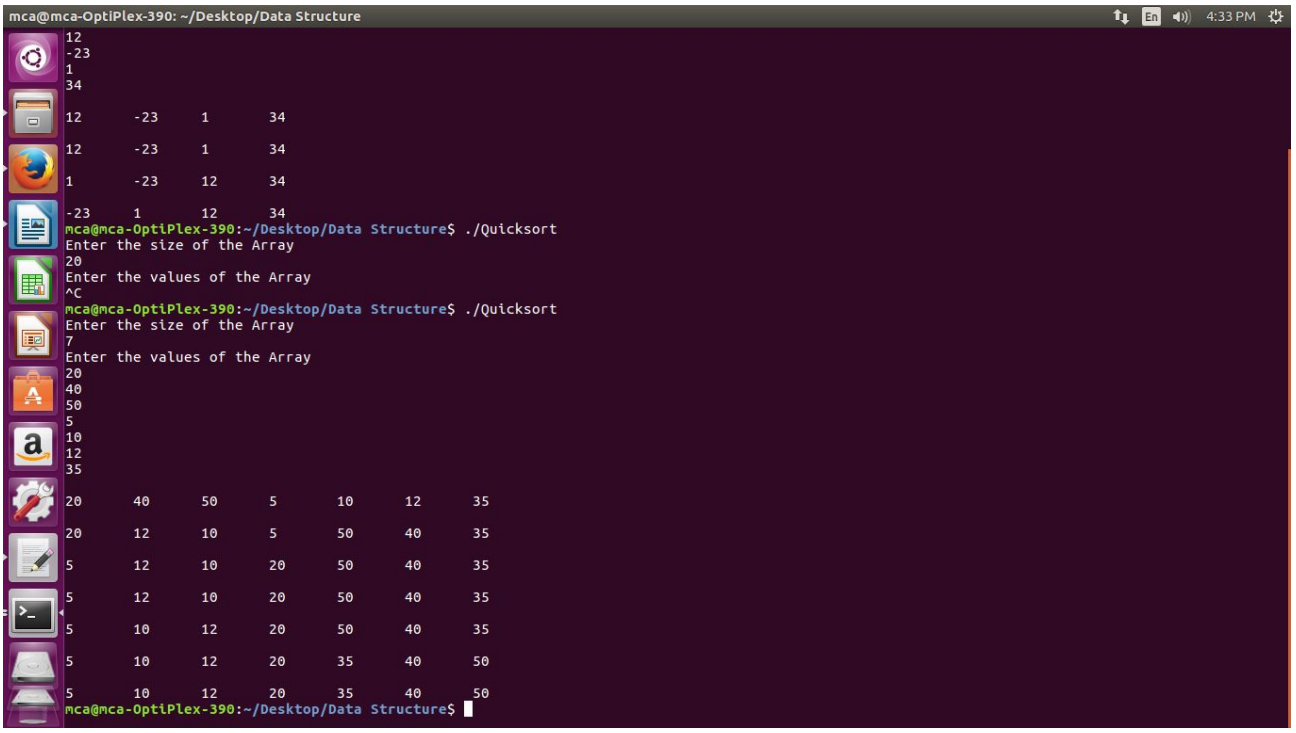
Code:-

```
#include<iostream>
#define max 25
using namespace std;


class QuickSort
{

	int n, A[max], lower, upper, temp, l, u, i;

	public:
		QuickSort()
		{
			n = 0;
			temp = 0;
			l = 0;
			u = 0;
			i = 0;
		}

		int getdata()
		{
			cout<<"Enter the size of the Array"<<endl;
			cin>>n;
			A[n];
			cout<<"Enter the values of the Array"<<endl;
			for(int i=0; i<n; i++)
			{
				cin>>A[i];
			}
		return n;
		}

		void display()
		{
			cout<<endl;
			for(int i=0; i<n; i++)
			{
				cout<<A[i]<<"\t";
			}
			cout<<endl;
		}


		void sort(int lower, int upper)
```

```
        {
                if ( lower < upper )
                {
                        int i = part ( A, lower, upper ) ;
                        sort ( lower, i - 1 ) ;
                        sort ( i + 1, upper ) ;
                }
        }

        int part ( int *a, int lower, int upper )
        {
                l = lower + 1 ;
                u = upper ;
                i = a[lower] ;
                while ( u >= l )
                {
                        while ( a[l] < i )
                        {
                                l++ ;
                        }
                        while ( a[u] > i )
                        {
                                u-- ;
                        }
                        if ( u > l )
                        {
                                temp = a[l] ;
                                a[l] = a[u] ;
                                a[u] = temp ;
                        }
                }
                temp = a[lower] ;
                a[lower] = a[u] ;
                a[u] = temp ;
                return u ;
        }
};

int main()
{

QuickSort qs;
int x = qs.getdata();
qs.display();
qs.sort(0, x-1);
cout<<"Sorted :";
qs.display();

return 0;
```

}

Screenshots:-



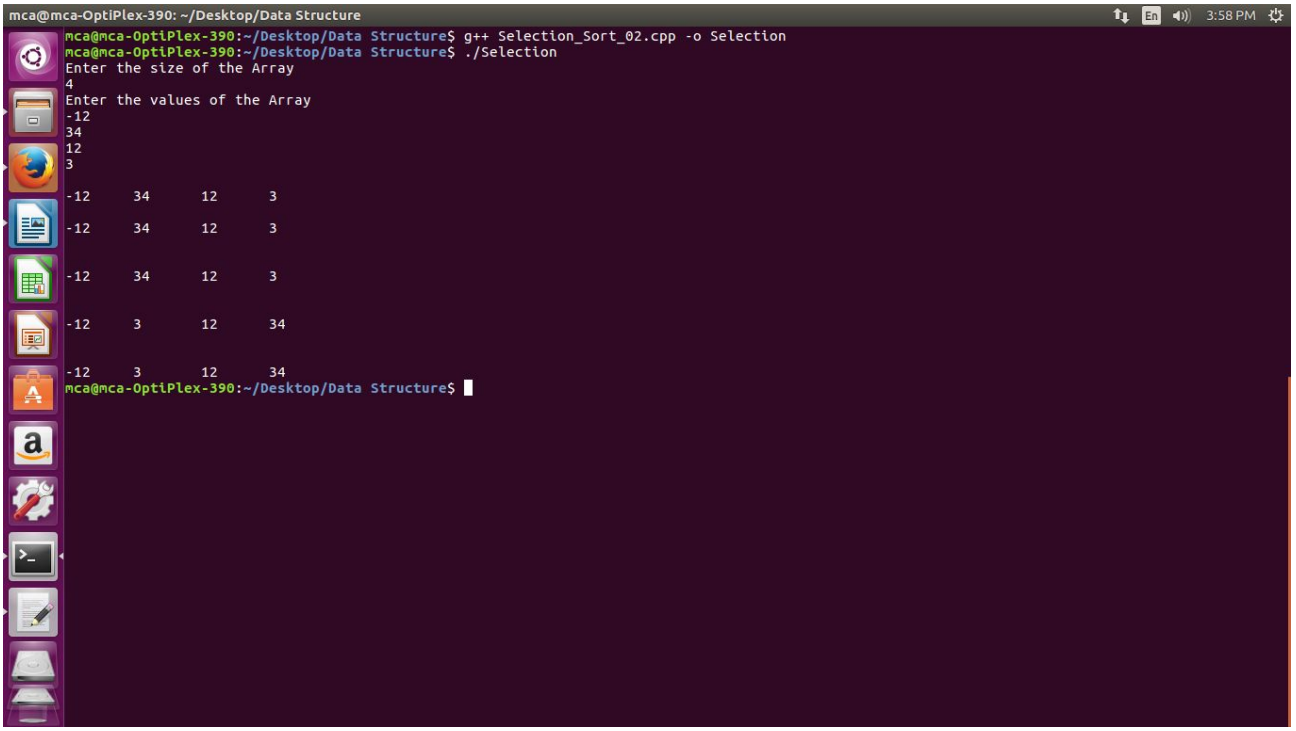
3. **Selection Sort**

Code:-

```
#include<iostream>
#define max 25
using namespace std;


class SelectionSort
{

	int n, A[max], temp, i, j, t, m;

	public:
		SelectionSort()
		{
			n = 0;
			m = 0;
			t = 0;
			i = 0;
			j = 0;
			temp = 0;
		}

		void getdata()
		{
			cout<<"Enter the size of the Array"<<endl;
			cin>>n;
			A[n];
			cout<<"Enter the values of the Array"<<endl;
			for(int i=0; i<n; i++)
			{
				cin>>A[i];
			}
		}

		void display()
		{
			cout<<endl;
			for(int i=0; i<n; i++)
			{
				cout<<A[i]<<"\t";
			}
			cout<<endl;
		}

		void sort()
		{
```

```
                for(i=0; i<n-1; i++)
                {
                        m = A[i];
                        t = i;
                        for(j=i+1; j<n; j++)
                        {
                                if(m>A[j])
                                {
                                        m = A[j];
                                        t = j;
                                }
                        }
                        display();
                        cout<<endl;
                        temp = A[i];
                        A[i] = A[t];
                        A[t] = temp;
                }

        }
};

int main()
{

SelectionSort ss;
ss.getdata();
ss.display();
ss.sort();
ss.display();

return 0;
}
```

Screenshots:-



4. **Radix Sort**

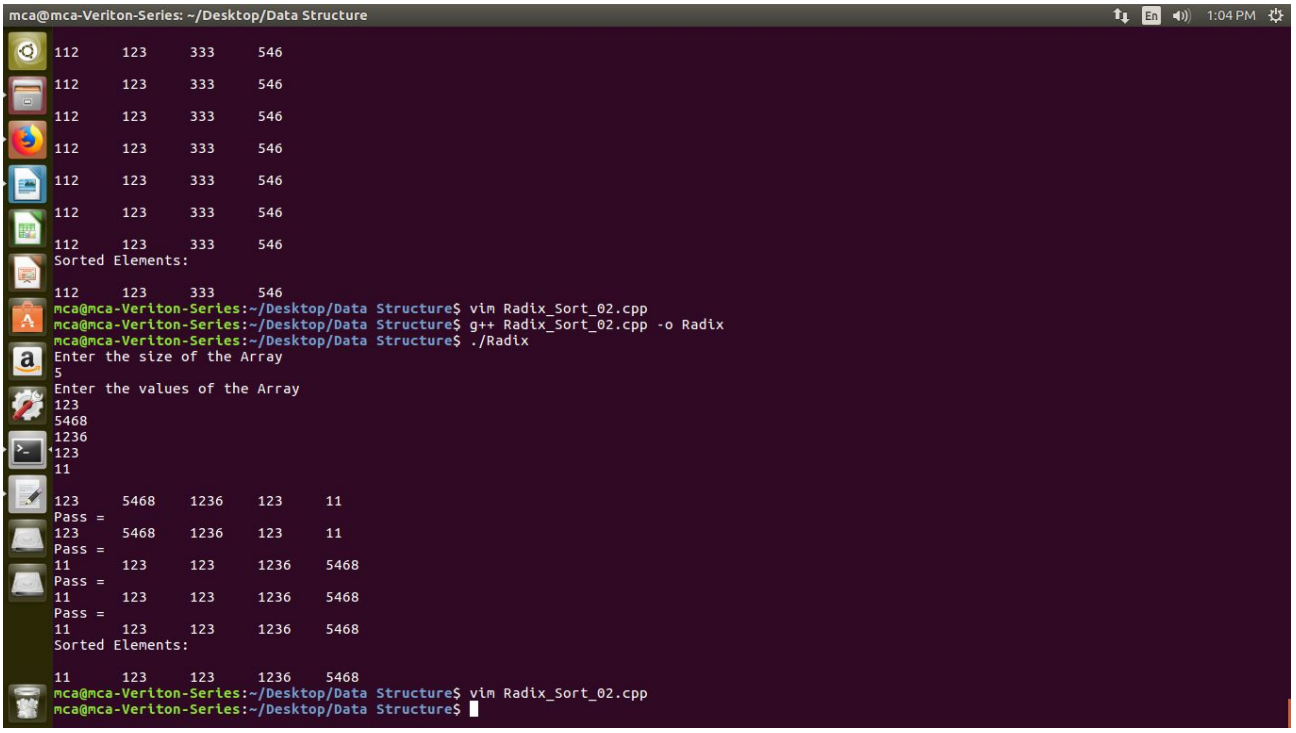
Code:-

```
#include<iostream>
#include<cmath>
#define m 25
using namespace std;

class Radix
{
 public:
        int n, A[m], max, x;

        Radix()
        {
                n=0; max=0; x=0;
        }

        void getdata()
                {
                        cout<<"Enter the size of the Array"<<endl;
                        cin>>n;
                        A[n];
                        cout<<"Enter the values of the Array"<<endl;
                        for(int i=0; i<n; i++)
                        {
```

```
                        cin>>A[i];
                }
        }

int getMax(int A[], int n)
{
        int temp=A[0];
        for(int i=0; i<n; i++)
        {
                if(A[i]>temp)
                {
                        temp = A[i];
                }
        }
        return temp;
}

void countsort(int A[], int n,int x)
{
        int count[10]={0}, i, output[n];

        // Counter Initialized
        // Exponential Game
        for(i=0; i<n; i++)
        {
                count[(A[i]/x)%10]++;
        }

        //Total Count
        for(i=1; i<10; i++)
        {
                count[i]=count[i]+count[i-1];
        }

        //Building Output Array
        for(i=n-1; i>=0; i--)
        {
                output[count[(A[i]/x)%10]-1] = A[i];
                count[(A[i]/x)%10]--;
        }
        // Copying the elements
        for(i=0; i<n; i++)
        {
                A[i] = output[i];
        }

}

void sort()
```

```
	{
		max=getMax(A, n);
		for(x=1; max/x>0; x=x*10)
		{
			cout<<"Pass = ";
			display();
			countsort(A, n, x);
		}
	}

	void display()
	{
		cout<<endl;
		for(int i=0; i<n; i++)
		{
			cout<<A[i]<<"\t";
		}
		cout<<endl;
	}
};

int main()
{
	Radix r;
	r.getdata();
	r.display();
	r.sort();
	cout<<"Sorted Elements: \n";
	r.display();
	return 0;
}
```

Screenshots:-



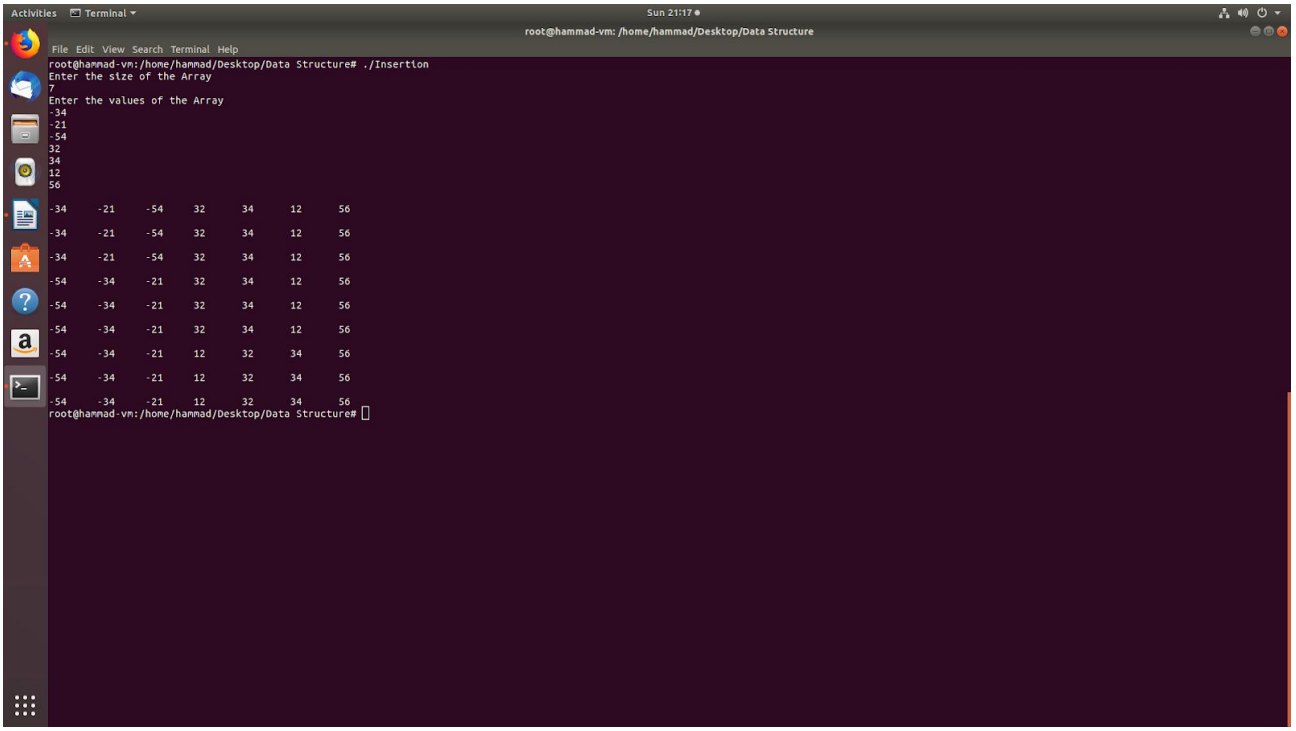
5. **Insertion Sort**

Code:-

```
#include<iostream>
#define max 50

using namespace std;
class InsertionSort
{
	int n, i, j, t, A[max];

	public:
		InsertionSort()
		{
			n = 0;
			i = 0;
			j = 0;
			t = 0;
		}

		int getdata()
		{
			cout<<"Enter the size of the Array"<<endl;
			cin>>n;
			A[n];
			cout<<"Enter the values of the Array"<<endl;
			for(int i=0; i<n; i++)
			{
				cin>>A[i];
			}
		}

		void display()
		{
			cout<<endl;
			for(int i=0; i<n; i++)
			{
				cout<<A[i]<<"\t";
			}
			cout<<endl;
		}


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

```
                {
                        t = A[i];
                        j = i-1;
                        while((t<A[j]) && (j>=0))
                        {
                                A[j+1] = A[j];
                                j = j-1;
                        }
                        A[j+1]=t;
                        display();
                }
        }

};

int main()
{

InsertionSort is;

is.getdata();
is.display();
is.sort();
is.display();

return 0;
}
```

Screenshots:-



6. **Shell Sort**

Code:-

```
#include<iostream>
#define max 25
using namespace std;


class ShellSort
{

        int n, A[max], temp, i, j;

        public:
                ShellSort()
                {
                        n = 0;
                        i = 0;
                        j = 0;
                        temp = 0;
                }

                void getdata()
                {
                        cout<<"Enter the size of the Array"<<endl;
                        cin>>n;
                        A[n];
```

```
                cout<<"Enter the values of the Array"<<endl;
                for(int i=0; i<n; i++)
                {
                        cin>>A[i];
                }
        }

        void display()
        {
                cout<<endl;
                for(int i=0; i<n; i++)
                {
                        cout<<A[i]<<"\t";
                }
                cout<<endl;
        }

        void sort()
        {
                for(i=n/2; i>0; i/=2)
                {
                        display();
                        for(j=i; j<n; j++)
                        {
                                for(int k=j-i; k >= 0; k = k-i)
                                {
                                        if(A[k+i] <= A[k] )
                                        {
                                                temp = A[k];
                                                A[k] = A[k+i];
                                                A[k+i] = temp;
                                        }
                                        else
                                        {       break; }
                                }
                        }
                }

        }
};

int main()
{

ShellSort ss;
ss.getdata();
ss.display();
ss.sort();
ss.display();
```

```
return 0;
}
```

Screenshots:-