```
#include<iostream>
using namespace std;
class SortPass
{
	private:
		int *arr;
		int size;
		int last;
		void buildHeap();
		void reHeapDown(int,int);
		void reHeapUp(int);
		void medianLeft(int,int);
	public:
		SortPass(int);
		void fillArray();
		void bubbleSort();
		void quickSort(int,int,int);
		void insertionSort();
		void shellSort();
		void selectionSort();
		void heapSort();
		void printArray();
};
SortPass::SortPass(int s)
{
	size=s;
	arr = new int[s];
```

```
	last=size-1;
}
void SortPass::fillArray()
{
	cout<<"PLease Enter the array elements"<<endl;
	for(int i=0;i<size;i++)
	{
		cout<<"Enter "<<i+1<<" element: ";
		cin>>arr[i];
	}
}
void SortPass::printArray()
{
	for(int i=0;i<size;i++)
	{
		cout<<arr[i]<<" ";
	}
}
void SortPass::bubbleSort()
{
	int current,walker,temp;
	bool sorted=false;
	current=0;
	cout<<"\n";
	while(current<=last && sorted==false)
	{
		walker=last;
```

```
                sorted=true;
                while(walker>current)
                {
                        if(arr[walker]<arr[walker-1])
                        {
                                sorted=false;
                                temp=arr[walker];
                                arr[walker]=arr[walker-1];
                                arr[walker-1]=temp;
                        }
                        walker=walker-1;
                }
                cout<<"\nAfter Pass "<<current+1<<" : ";
                printArray();
                current++;
        }
}
void SortPass::quickSort(int l,int r,int i)
{
        int pivot,left,right,sortLeft,sortRight,temp;
        left=l;
        right=r;
        medianLeft(left,right);
        pivot=arr[left];
        sortLeft=left+1;
        sortRight=right;
        cout<<"\n";
```

```
while(sortLeft<=sortRight)
{
	while(arr[sortLeft]<pivot)
	{
		sortLeft++;
	}
	while(arr[sortRight]>=pivot)
	{
		sortRight--;
	}
	if(sortLeft<=sortRight)
	{
		temp=arr[sortLeft];
		arr[sortLeft]=arr[sortRight];
		arr[sortRight]=temp;
		sortLeft++;
		sortRight--;
	}
}
arr[left]=arr[sortLeft-1];
arr[sortLeft-1]=pivot;
i++;
cout<<"\nAfter Pass "<<i<<" : ";
printArray();
if(left<sortRight)
{
	quickSort(left,sortRight-1,i++);
```

```
        }
        if(sortLeft<right)
        {
                quickSort(sortLeft,right,i++);
        }
}

void SortPass::medianLeft(int left,int right)
{
        int temp;
        int mid=(left+right)/2;
        if(arr[left]>arr[mid])
        {
                temp=arr[left];
                arr[left]=arr[mid];
                arr[mid]=temp;
        }
        if(arr[left]>arr[right])
        {
                temp=arr[left];
                arr[left]=arr[right];
                arr[right]=temp;
        }
        if(arr[mid]>arr[right])
        {
                temp=arr[mid];
                arr[mid]=arr[right];
```

```
                arr[right]=temp;
        }
        temp=arr[left];
        arr[left]=arr[mid];
        arr[mid]=temp;
}

void SortPass::insertionSort()
{
        int hold,current,walker;
        current=1;
        cout<<"\n";
        while(current<=last)
        {
                hold=arr[current];
                walker=current-1;
                while(walker>=0 && hold<arr[walker])
                {
                        arr[walker+1]=arr[walker];
                        walker--;
                }
                arr[walker+1]=hold;
                cout<<"\nAfter Pass "<<current<<" : ";
                printArray();
                current++;
        }
}
```

```
void SortPass::shellSort()
{
	int incre,current,hold,walker,i;
	incre=last/2;
	i=1;
	cout<<"\n";
	while(incre!=0)
	{
		current=incre;
		while(current<=last)
		{
			hold=arr[current];
			walker=current-incre;
			while(walker>=0 && hold<arr[walker])
			{
				arr[walker+incre]=arr[walker];
				walker=walker-incre;
			}
			arr[walker+incre]=hold;
			current=current+1;
		}
		cout<<"\nAfter Pass "<<i<<" : ";
		i++;
		printArray();
		incre=incre/2;
	}
}
```

```
void SortPass::selectionSort()
{
	int i, j, min;
	cout<<"\n";
	for(i = 0; i<size-1; i++)
	{
		min = i;
		for(j = i+1; j<size; j++)
		{
		if(arr[j] < arr[min])
			{
				min = j;
			}
			int temp=arr[i];
			arr[i]=arr[min];
			arr[min]=temp;
		}
		cout<<"\nAfter Pass "<<i+1<<" : ";
		printArray();
  }
}
void SortPass::heapSort()
{
	int sorted,holdData,i;
	i=1;
	buildHeap();
	sorted=last;
```

```
	cout<<"\n";
	while(sorted>0)
	{
		holdData=arr[0];
		arr[0]=arr[sorted];
		arr[sorted]=holdData;
		sorted--;
		reHeapDown(0,sorted);
		cout<<"\nAfter Pass "<<i<<" : ";
		printArray();
	}
}

void SortPass::buildHeap()
{
	cout<<"\n";
	for(int walker=1;walker<=last;walker++)
	{
		reHeapUp(walker);
	}
	cout<<"\nArray in Heap Format : ";
	printArray();
	cout<<"\n";
}

void SortPass::reHeapUp(int newNode)
{
```

```
	if(newNode!=0)
	{
		int temp;
		int parent=(newNode-1)/2;
		if(arr[newNode]>arr[parent])
		{
			temp=arr[newNode];
			arr[newNode]=arr[parent];
			arr[parent]=temp;
			reHeapUp(parent);
		}
	}
}

void SortPass::reHeapDown(int root,int lastIndex)
{
	int leftKey,rightKey,largeChildKey,largeChildIndex,lowKey;
	leftKey=rightKey=largeChildKey=largeChildIndex=lowKey=0;
	if((root*2)+1<=lastIndex)
	{
		leftKey=arr[(root*2)+1];
		if((root*2)+2<=lastIndex)
		{
			rightKey=arr[(root*2)+2];
		}
		else
		{
```

```
                rightKey=lowKey;
            }
            if(leftKey>rightKey)
            {
                largeChildKey=leftKey;
                largeChildIndex=(root*2)+1;
            }
            else
            {
                largeChildKey=rightKey;
                largeChildIndex=(root*2)+2;
            }
            if(arr[root]<arr[largeChildIndex])
            {
                int temp=arr[root];
                arr[root]=arr[largeChildIndex];
                arr[largeChildIndex]=temp;
                reHeapDown(largeChildIndex,lastIndex);
            }
        }
}
int main()
{
        SortPass bubbleobj(5),quickobj(10),insertionobj(5),shellobj(10),selectobj(5),heapobj(5);

        cout<<"Bubble Sort"<<endl<<endl;
        bubbleobj.fillArray();
```

```
cout<<"\nBefore Bubble Sort elements are"<<endl;
bubbleobj.printArray();
bubbleobj.bubbleSort();
cout<<"\n\nAfter Bubble Sort elements are"<<endl;
bubbleobj.printArray();

cout<<"\n\nQuick Sort"<<endl<<endl;
quickobj.fillArray();
cout<<"\nBefore Quick Sort elements are"<<endl;
quickobj.printArray();
quickobj.quickSort(0,11,0);
cout<<"\n\nAfter Quick Sort elements are"<<endl;
quickobj.printArray();

cout<<"\n\nInsertion Sort"<<endl<<endl;
insertionobj.fillArray();
cout<<"\nBefore Insertion Sort elements are"<<endl;
insertionobj.printArray();
insertionobj.insertionSort();
cout<<"\n\nAfter Insertion Sort elements are"<<endl;
insertionobj.printArray();

cout<<"\n\nShell Sort"<<endl<<endl;
shellobj.fillArray();
cout<<"\nBefore Shell Sort elements are"<<endl;
shellobj.printArray();
shellobj.shellSort();
```

```
        cout<<"\n\nAfter Shell Sort elements are"<<endl;
        shellobj.printArray();

        cout<<"\n\nSelection Sort"<<endl<<endl;
        selectobj.fillArray();
        cout<<"\nBefore Selection Sort elements are"<<endl;
        selectobj.printArray();
        selectobj.selectionSort();
        cout<<"\n\nAfter Selection Sort elements are"<<endl;
        selectobj.printArray();

        cout<<"\n\nHeap Sort"<<endl<<endl;
        heapobj.fillArray();
        cout<<"\nBefore Heap Sort elements are"<<endl;
        heapobj.printArray();
        heapobj.heapSort();
        cout<<"\n\nAfter Heap Sort elements are"<<endl;
        heapobj.printArray();
        return 0;
}
```

C:\Users\Vaidhai\Desktop\assignment6.exe

```
Bubble Sort

PLease Enter the array elements
Enter 1 element: 4
Enter 2 element: 48
Enter 3 element: 3
Enter 4 element: 98
Enter 5 element: 45

Before Bubble Sort elements are
4 48 3 98 45

After Pass 1 : 3 4 48 45 98
After Pass 2 : 3 4 45 48 98
After Pass 3 : 3 4 45 48 98

After Bubble Sort elements are
3 4 45 48 98

Quick Sort

PLease Enter the array elements
Enter 1 element: 56
Enter 2 element: 4
Enter 3 element: 85
Enter 4 element: 96
Enter 5 element: 78
Enter 6 element: 8
Enter 7 element: 23
Enter 8 element: 12
Enter 9 element: 45
Enter 10 element: 856

Before Quick Sort elements are
56 4 85 96 78 8 23 12 45 856

After Pass 1 : 0 4 8 96 78 85 23 12 45 856

After Pass 2 : 0 4 8 96 78 85 23 12 45 856

After Pass 3 : 0 4 8 23 45 12 56 85 78 856

After Pass 4 : 0 4 8 12 23 45 56 85 78 856

After Pass 5 : 0 4 8 12 23 45 56 85 78 856

After Pass 5 : 0 4 8 12 23 45 56 85 78 96

After Pass 6 : 0 4 8 12 23 45 56 78 85 96
```

C:\Users\Vaidhai\Desktop\assignment6.exe

```
After Pass 6 : 0 4 8 12 23 45 56 78 85 96

After Pass 7 : 0 4 8 12 23 45 56 78 85 96

After Quick Sort elements are
0 4 8 12 23 45 56 78 85 96

Insertion Sort

PLease Enter the array elements
Enter 1 element: 4
Enter 2 element: 85
Enter 3 element: 41
Enter 4 element: 2
Enter 5 element: 32

Before Insertion Sort elements are
4 85 41 2 32

After Pass 1 : 4 85 41 2 32
After Pass 2 : 4 41 85 2 32
After Pass 3 : 2 4 41 85 32
After Pass 4 : 2 4 32 41 85

After Insertion Sort elements are
2 4 32 41 85

Shell Sort

PLease Enter the array elements
Enter 1 element: 5
Enter 2 element: 65
Enter 3 element: 45
Enter 4 element: 12
Enter 5 element: 20
Enter 6 element: 9
Enter 7 element: 6
Enter 8 element: 75
Enter 9 element: 88
Enter 10 element: 52

Before Shell Sort elements are
5 65 45 12 20 9 6 75 88 52

After Pass 1 : 5 9 6 12 20 52 45 75 88 65
After Pass 2 : 5 9 6 12 20 52 45 65 88 75
After Pass 3 : 5 6 9 12 20 45 52 65 75 88
```

C:\Users\Vaidhai\Desktop\assignment6.exe

```
After Shell Sort elements are
5 6 9 12 20 45 52 65 75 88

Selection Sort

PLease Enter the array elements
Enter 1 element: 85
Enter 2 element: 8
Enter 3 element: 45
Enter 4 element: 1
Enter 5 element: 21

Before Selection Sort elements are
85 8 45 1 21

After Pass 1 : 21 85 8 45 1
After Pass 2 : 21 1 85 8 45
After Pass 3 : 21 1 45 85 8
After Pass 4 : 21 1 45 8 85

After Selection Sort elements are
21 1 45 8 85

Heap Sort

PLease Enter the array elements
Enter 1 element: 78
Enter 2 element: 45
Enter 3 element: 12
Enter 4 element: 89
Enter 5 element: 56

Before Heap Sort elements are
78 45 12 89 56

Array in Heap Format : 89 78 12 45 56


After Pass 1 : 78 56 12 45 89
After Pass 1 : 56 45 12 78 89
After Pass 1 : 45 12 56 78 89
After Pass 1 : 12 45 56 78 89

After Heap Sort elements are
12 45 56 78 89
--------------------------------
Process exited after 63.51 seconds with return value 0
Press any key to continue . . .
```