

```

1: //
   Assignment no-9
2: // Name- Prerana Rajesh Gajare      Class-SEIT
   RollNo-SI41
3: /*PROBLEM STATEMENT:-
4:      Implement Heap sort to sort a given
   set of values using max or min-heap.
5: */
6: //Source Code:-
7: #include <iostream>
8: using namespace std;
9:
10: //Class hs
11: class hs
12: {
13:     public:
14:         //Function declaration
15:         void getdata(int[],int);
16:         void max_heapify(int [],int,int);
17:         void min_heapify(int[],int,int);
18:         void heapsort(int[],int,int);
19:         void display(int [],int);
20: };
21:
22: // TO ACCEPT DATA IN THE FORM OF ARRAY
23: void hs :: getdata(int heap[], int n)
24: {
25:     cout<<"Enter the elements:";
26:     for (int i = 0; i < n; i++)
27:     {
28:         cin >> heap[i];
29:     }
30: }
31:

```

```

32: // TO PERFORM MAX HEAP OPERATION
33: void hs :: max_heapify(int heap[],int n,int i)
34: {
35:     int largest=i;// Initialize largest as root
36:     int left=2*i+1;//Position of left child
37:     int right=2*i+2;//Position of right child
38:
39:     // If left child is greater than root ,set
position of left as largest
40:     if (left<n && heap[left]>heap[largest])
41:         largest=left;
42:     else
43:         largest=i;
44:
45:     // If right child is greater than root ,set
position of right as largest
46:     if(right<n && heap[right]>heap[largest])
47:         largest=right;
48:
49:     // If largest is not root
50:     if(largest!=i)
51:     {
52:
53:         swap(heap[i],heap[largest]);
54:
55:         // Recursively heapify the affected sub-
tree
56:         max_heapify(heap,n,largest);
57:     }
58:
59: }
60:
61: // TO PERFORM MIN HEAP OPERATION
62: void hs :: min_heapify(int heap[], int n,int i)

```

```

63: {
64:
65:     int smallest = i; // Initialize smallest as
    root
66:     int left=2*i+1; //Position of left child
67:     int right=2*i+2; //Position of right child
68:
69:     // If left child is smaller than root, set
    position of left as smallest
70:     if (left < n && heap[left]<heap[smallest])
71:         smallest = left;
72:     else
73:         smallest=i;
74:
75:     // If right child is smaller than root ,set
    position of right as smallest
76:     if (right < n && heap[right]<heap[smallest])
77:         smallest = right;
78:
79:     // If smallest is not root
80:     if (smallest != i) {
81:         swap(heap[i], heap[smallest]);
82:
83:         // Recursively heapify the affected sub-
        tree
84:         min_heapify(heap, n, smallest);
85:     }
86:
87: }
88:
89: // TO PERFORM HEAPSORT OPERATION
90: void hs :: heapsort(int heap[], int n, int m)
91: {
92:     // Building heap

```

```

93:     for (int i=n/2-1;i>= 0;i--)
94:     {
95:         switch(m)
96:         {
97:             case 1:
98:                 max_heapify(heap, n, i);
99:                 break;
100:            case 2:
101:                min_heapify(heap,n,i);
102:                break;
103:        }
104:
105:    }
106:
107:    // To extract one by one element from heap
108:    for (int i=n-1; i>=0;i--)
109:    {
110:        //Swap the first and last index
111:        swap(heap[0], heap[i]);
112:        switch(m)
113:        {
114:            case 1:
115:                max_heapify(heap, i, 0);
116:                break;
117:            case 2:
118:                min_heapify(heap,i,0);
119:                break;
120:        }
121:
122:    }
123:
124: }
125:
126: // TO DISPLAY THE DATA IN FORM OF ARRAY

```

```

127: void hs :: display(int heap[], int n)
128: {
129:     for (int i=0; i<n;i++)
130:     {
131:         cout<<"\t"<< heap[i];
132:     }
133: }
134:
135:
136: int main()
137: {
138:
139:     int n;
140:     int heap[15];//Declaring an array heap of size
15
141:     hs h;// Creating object of class hs
142:
143:     // Input the number of elements to be present
in array.
144:     cout<<"Enter the no of elements to be present
in array:";
145:     cin>>n;
146:
147:     int l;
148:     do
149:     {
150:         cout<<"\nEnter the opertion to be
performed:";
151:         cout<<"\n1)Insert";
152:         cout<<"\n2)Display";
153:         cout<<"\n3)Heap Sort";
154:         cout<<"\n4)Exit";
155:         cout<<"(1,2,3,4):";
156:         cin>>l;

```

```

157:         switch(1)
158:         {
159:             case 1:
160:                 h.getdata(heap,n);//Calling
getdata function
161:                 break;
162:             case 2:
163:                 h.display(heap, n);//Calling
display function
164:                 break;
165:             case 3:
166:                 int m;
167:                 do
168:                 {
169:                     cout<<"\nEnter the
poeration to be performed:";
170:                     cout<<"\n1)Max heap";
171:                     cout<<"\n2)Min heap";
172:                     cout<<"\n3)Exit";
173:                     cout<<"\n(1,2,3):";
174:                     cin>>m;
175:                     switch(m)
176:                     {
177:                         case 1:
178:
179:                             h.heapsort(heap,n,
m);//Calling heapsort function to perform maxheap
180:                             cout<<"\nSorted
heap:";
181:                             h.display(heap,
n);//Display the elements of max heap
182:                             break;
183:                         case 2:
184:

```

```

185:                                     h.heapsort(heap,n,
    m);//Calling heapsort function to perform minheap
186:                                     cout<<"\nSorted
heap:";
187:                                     h.display(heap,
n);//Display the elements of minheap
188:                                     break;
189:                             case 3:
190:                                 cout<<"The End";
191:                                 break;
192:                             default:
193:                                 cout<<"Wrong
choice";
194:                                     }
195:                             }while(m!=3);
196:                             break;
197:                         case 4:
198:                             cout<<"The End";
199:                             break;
200:                         default:
201:                             cout<<"Wrong Choice";
202:                     }
203:                 }while(l!=4);
204:
205:     }

```

Enter the no of elements to be present in array:6

Enter the operation to be performed:

- 1)Insert
- 2)Display
- 3)Heap Sort
- 4)Exit(1,2,3,4):1

Enter the elements:

3
9
2
1
4
5

Enter the operation to be performed:

- 1)Insert
- 2)Display
- 3)Heap Sort
- 4)Exit(1,2,3,4):2

3 9 2 1 4 5

Enter the operation to be performed:

- 1)Insert
- 2)Display
- 3)Heap Sort
- 4)Exit(1,2,3,4):3

Enter the operation to be performed:

- 1)Insert
- 2)Display
- 3)Heap Sort
- 4)Exit(1,2,3,4):3

Enter the poeration to be performed:

- 1)Max heap
- 2)Min heap
- 3)Exit
- (1,2,3):1

Sorted heap: 1 2 3 4 5 9

Enter the poeration to be performed:

- 1)Max heap
- 2)Min heap
- 3)Exit
- (1,2,3):2

Sorted heap: 9 5 4 3 2 1

Enter the poeration to be performed:

- 1)Max heap
- 2)Min heap
- 3)Exit
- (1,2,3):3

The End

Enter the operation to be performed:

3)Exit

(1,2,3):1

Sorted heap: 1 2 3 4 5 9

Enter the poeration to be performed:

1)Max heap

2)Min heap

3)Exit

(1,2,3):2

Sorted heap: 9 5 4 3 2 1

Enter the poeration to be performed:

1)Max heap

2)Min heap

3)Exit

(1,2,3):3

The End

Enter the opertion to be performed:

1)Insert

2)Display

3)Heap Sort

4)Exit(1,2,3,4):4

The End

Process exited after 70.8 seconds with return value 0

Press any key to continue . . .