

List of Experiment

Subject: CS103 Data Structures

Faculty : Ashutosh Gupta

Experiment No.	Experiment Name
1	Write a program to search an element using linear search
2	Write a program to search an element using Binary search
3	Write a program to sort the list using Insertion Sort.
4	Write a program to sort the list using Bubble Sort.
5	Write a program to sort the list using Selection Sort.
6	Write a program to sort the list using Shell Sort.
7	Write a program to sort the list using Quick Sort.
8	Write a program to sort the list using Merge Sort.
9	Write a program to sort the list using Radix Sort.
10	Write a program to merge two sorted list.
11	Write a program to understand the concept of Pointer.
12	Write a program to Perform the basic operation of Stack using array / linked List.
13	Write a program to Perform the basic operation of Queue using array / linked List.
14	Write a program to Perform the basic operation of Circular Queue using array / linked List.
15	Write a program to Perform the basic operation of Linked List.
16	Write a program to Perform the basic operation of circular linked List.
17	Write a program to Perform the basic operation of doubly linked List.
18	Write a program to traverse binary tree..
19	Write a program to Perform the concept of recursion.
20	Write a program to sort the list using Heap sort.

Program 1

Date : / /

Aim : Write a program to search an element using linear search

Algorithm:

LINEAR(DATA,N, ITEM,LOC)

1. Set DATA[N+1] = ITEM
2. Set LOC = 1
3. Repeat while DATA[LOC] !=ITEM
Set LOC = LOC +1
4. If LOC = N+1 , then Set LOC =0
5. Exit.

Source code:

Output:

Enter item :23
Search Successful, 23 found at location 4.

Complexity:

Algorithm	Worst case
Linear Search	O(n)

Selected Questions :

- Explain linear search with an example.
- When does interpolation search performs better than binary search?
- Distinguish between sorting and searching.

Program 2

Date : / /

Aim : Write a program to search an element using Binary search

Algorithm:

BINARY(DATA, LB, UB, ITEM, LOC)

1. Set BEG=LB, END=UB, and MID= INT((BEG+END)/2)
2. Repeat Steps 3 and 4 while BEG<=END and DATA[MID]!=ITEM
3. if ITEM<DATA[MID] then
 set END=MID – 1
 else
 set BEG = MID +1
4. Set MID = = INT((BEG+END)/2)
5. if DATA[MID]=ITEM, then
 set LOC = MID
 else
 set LOC = NULL
6. Exit.

Source code:

Output:

Enter item :23
Search Successful, 23 found at location 4.

Complexity:

Algorithm	Worst case
Binary Search	$O(\log_2 n)$

Selected Questions :

1. Find the time complexity to perform binary search in an unsorted array of 'n' numbers.
2. Explain how the divide and conquer technique is implemented in binary search.
3. List the advantages of binary search.
4. Why searching is easier in sorted list.
Compare linear and binary search
5. Discuss the merits and demerits of binary search algorithm

Program 3

Date : / /

Aim : Write a program to sort the list using Insertion Sort.

Algorithm:

```
(Insertion Sort) INSERTION(A, N).
This algorithm sorts the array A with N elements.
1. Set A[0] := -∞. [Initializes sentinel element.]
2. Repeat Steps 3 to 5 for K = 2, 3, ..., N:
3.   Set TEMP := A[K] and PTR := K - 1.
4.   Repeat while TEMP < A[PTR]:
      (a) Set A[PTR + 1] := A[PTR]. [Moves element forward.]
      (b) Set PTR := PTR - 1.
      [End of loop.]
5.   Set A[PTR + 1] := TEMP. [Inserts element in proper place.]
      [End of Step 2 loop.]
6. Return.
```

Source code:

Output:

Enter list : 77 33 44 11 88 22 66 55
Sorted list : 11 22 33 44 55 66 77 88

Complexity:

Algorithm	Worst case	Average Case
Insertion Sort	$O(n^2)$	$O(n^2)$

Selected Questions :

1. What are the various kinds of sorting techniques? Which is has best case?
2. Give the difference between external and internal sorting. Give names of some external sorting techniques.
3. Write the average and best case analysis of insertion sort

Program 4

Date : / /

Aim : Write a program to sort the list using Bubble Sort.

Algorithm:

1. Repeat step 2 and 3 for K=1 to N-1
2. set PTR = 1
3. Repeat while PTR <=N-K
 - (a) if DATA[PTR] > DATA[PTR + 1] then
Interchange DATA[PTR] and DATA [PTR =1]
[End of If structure.]
 - (b) set PTR =PTR +1
[End of Inner loop]
[End of Step1 Outer loop.]
4. Exit.

Source code:

Output:

Enter list :77 33 44 11 88 22 66 55
Sorted list : 11 22 33 44 55 66 77 88

Complexity:

Algorithm	Worst case	Average Case
Bubble	$O(n^2)$	$O(n^2)$

Selected Questions :

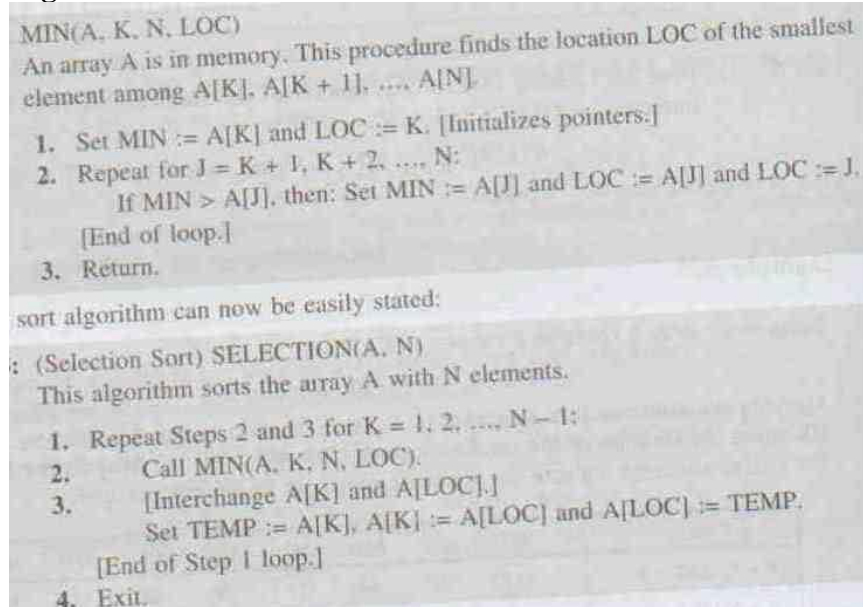
1. What is the time complexity of bubble sort?
2. What are the factors to be considered while choosing a sorting technique?
3. What is the total number of comparisons and total number of exchanges in Bubble Sort for the worst case situation?
4. What is the need for external sorting?

Program 5

Date : / /

Aim : Write a program to sort the list using Selection Sort.

Algorithm:



Source code:

Output:

Enter list : 77 33 44 11 88 22 66 55
Sorted list : 11 22 33 44 55 66 77 88

Complexity:

Algorithm	Worst case	Average Case
Selection	$O(n^2)$	$O(n^2)$

Selected Questions :

- a. Compare above three methods of sorting for ideal, worst and average cases.

Program 6

Date : / /

Aim : Write a program to sort the list using Shell Sort.

Algorithm:

```
do {
    do{
        swap=0;
        for(i=0;i<size-gap;i++)
        {
            If (a[i] > a[i+gap])
            {
                Temp=a[i];
                a[i] = a[i+gap])
                a[i+gap]=temp;
                swap=1
            }
        }
    }while(swap);
} while(gap=gap/2);
```

Source code:

Output:

```
Enter list      :77 33 44 11 88 22 66 55
Sorted list    : 11 22 33 44 55 66 77 88
```

Complexity:

Selected Questions :

Program 7

Date : / /

Aim : Write a program to sort the list using Quick Sort.

Algorithm:

QUICK(A,N,BEG,END,LOC)

1. Set LEFT=BEG, RIGHT=END and LOC=BEG
2. (a) Repeat while A[LOC] != A[RIGHT] and LOC != RIGHT
RIGHT = RIGHT -1
- (b) If LOC = RIGHT, then : Return
- (c) If A[LOC] > A[RIGHT] then
 - (i) TEMP= A[LOC], A[LOC]=A[RIGHT], A[RIGHT]=TEMP
 - (ii) Set LOC =RIGHT
 - (iii) Go to Step 3.
3. (a) Repeat while A[LEFT] != A[LOC] and LEFT != LOC
LEFT = LEFT +1
- (b) If LOC = LEFT, then : Return
- (c) If A[LEFT] > A[LOC] then
 - (i) TEMP= A[LOC], A[LOC]=A[LEFT], A[LEFT]=TEMP
 - (ii) Set LOC =LEFT
 - (iii) Go to Step 3.

QUICKSORT(A,N)

1. TOP = NULL
2. If N>1 then TOP=TOP+1, LOWER[1]=1, UPPER[1]=N
3. Repeat steps 4 to 7 while TOP != NULL
4. Set BEG=LOWER[TOP], END = UPPER[TOP],TOP=TOP-1
5. Call QUICK(A,N,BEG,END,LOC)
6. if BEG < LOC - 1, then
TOP=TOP +1, LOWER[TOP] = BEG, UPPER[TOP]=LOC-1
7. If LOC + 1 < END, then
TOP=TOP+1, LOWER[TOP]=LOC+1, UPPER[TOP]=END
8. Exit.

Source code:

Output:

Enter list :77 33 44 11 88 22 66 55
Sorted list : 11 22 33 44 55 66 77 88

Complexity:

Algorithm	Worst case	Average Case
Quick sort	$O(n^2)$	$O(n \log n)$

Selected Questions :

- Write the best, worst and average case time complexity estimates of Quick Sort algorithm.
- Which is the best way of choosing the pivot element in quick sort?

Program 8

Date : / /

Aim : Write a program to sort the list using Merge Sort.

Algorithm:

```
MERGEPAASS(A, N, L, B)
The N-element array A is composed of sorted subarrays where each subarray
has L elements except possibly the last subarray, which may have fewer than L
elements. The procedure merges the pairs of subarrays of A and assigns them
to the array B.
1. Set  $Q := \text{INT}(N/(2*L))$ ,  $S := 2*L*Q$  and  $R := N - S$ .
2. [Use Procedure 9.5 to merge the Q pairs of subarrays.]
   Repeat for  $J = 1, 2, \dots, Q$ :
     (a) Set  $LB := 1 + (2*J - 2)*L$ . [Finds lower bound of first array.]
     (b) Call MERGE(A, L, LB, A, L, LB + L, B, LB).
   [End of loop.]
3. [Only one subarray left?]
   If  $R \leq L$ , then:
     Repeat for  $J = 1, 2, \dots, R$ :
       Set  $B(S + J) := A(S + J)$ .
     [End of loop.]
   Else:
     Call MERGE(A, L, S + 1, A, R, L + S + 1, B, S + 1).
   [End of If structure.]
4. Return.

MERGESORT(A, N)
This algorithm sorts the N-element array A using an auxiliary array B.
1. Set  $L := 1$ . [Initializes the number of elements in the subarrays.]
2. Repeat Steps 3 to 6 while  $L < N$ :
3.   Call MERGEPAASS(A, N, L, B).
4.   Call MERGEPAASS(B, N, 2 * L, A).
5.   Set  $L := 4 * L$ .
   [End of Step 2 loop.]
6. Exit.
```

Source code:

Output:

Enter list : 77 33 44 11 88 22 66 55

Sorted list : 11 22 33 44 55 66 77 88

Complexity:

Algorithm	Worst case	Average Case
Merge sort	$O(n \log n)$	$O(n \log n)$

Selected Questions :

- Merge sort is better than insertion sort. Why?

Program 9

Date : / /

Aim : Write a program to sort the list using Radix Sort.

Algorithm:

Source code:

Output:

Enter list : 77 33 44 11 88 22 66 55
Sorted list : 11 22 33 44 55 66 77 88

Complexity:

Selected Questions :

1. What is the factor that is involved in radix sort?

Program 10

Date : / /

Aim : Write a program to merge two sorted list.

Algorithm:

```
MERGING(A, R, B, S, C)
Let A and B be sorted arrays with R and S elements, respectively. This algorithm
merges A and B into an array C with N = R + S elements.
1. [Initialize.] Set NA := 1, NB := 1 and PTR := 1.
2. [Compare.] Repeat while NA ≤ R and NB ≤ S:
    If A[NA] < B[NB], then:
        (a) [Assign element from A to C.] Set C[PTR] := A[NA].
        (b) [Update pointers.] Set PTR := PTR + 1 and NA := NA + 1.
    Else:
        (a) [Assign element from B to C.] Set C[PTR] := B[NB].
        (b) [Update pointers.] Set PTR := PTR + 1 and NB := NB + 1.
    [End of If structure.]
    [End of loop.]
3. [Assign remaining elements to C.]
    If NA > R, then:
        Repeat for K = 0, 1, 2, ..., S - NB:
            Set C[PTR + K] := B[NB + K].
        [End of loop.]
    Else:
        Repeat for K = 0, 1, 2, ..., R - NA:
            Set C[PTR + K] := A[NA + K].
        [End of loop.]
    [End of If structure.]
4. Exit.
```

Source code:

Output:

Enter Array A : 22 44 55 77 88

Enter Array B : 33 66 70

Merged array : 22 33 44 55 66 70 77 88

Complexity:

Selected Questions :

Program 11

Date : / /

Aim : Write a program to understand the concept of Pointer.

Algorithm:

Source code:

Output:

Complexity:

Selected Questions :

- Differentiate between static and dynamic storage with the help of an example.
- how the structure can be accessed by a pointer variable ?
- Discuss the representations of pointer variable in memory with suitable examples
- Compare linked allocation with sequential allocation.

Program 12

Date : / /

Aim : Write a program to Perform the basic operation of Stack using array / linked List.

Algorithm:

PUSH(STACK, TOP, MAXSTK, ITEM)

This procedure pushes an ITEM onto a stack.

1. [Stack already filled?]
If $TOP = MAXSTK$, then: Print: OVERFLOW, and Return.
2. Set $TOP := TOP + 1$. [Increases TOP by 1.]
3. Set $STACK[TOP] := ITEM$. [Inserts ITEM in new TOP position.]
4. Return.

POP(STACK, TOP, ITEM)

This procedure deletes the top element of STACK and assigns it to the variable ITEM.

1. [Stack has an item to be removed?]
If $TOP = 0$, then: Print: UNDERFLOW, and Return.
2. Set $ITEM := STACK[TOP]$. [Assigns TOP element to ITEM.]
3. Set $TOP := TOP - 1$. [Decreases TOP by 1.]
4. Return.

Source code:

Output:

Selected Questions :

- Define stack and give some applications of stacks.
- Write the role of stack in function call.

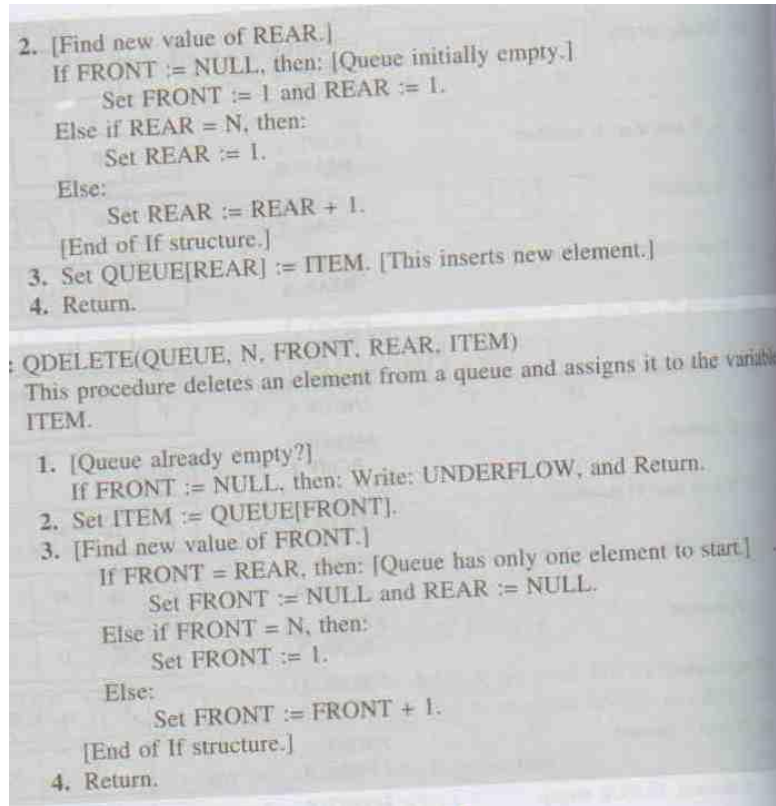
Program 13

Date : / /

Aim : Write a program to Perform the basic operation of Queue using array / linked List.

Algorithm:

1. **If FRONT =1 and REAR=N or if FRONT=REAR+1 then :
Print : Overflow and return**



Source code:

Output:

Selected Questions :

- What is queue? Give examples.
- What is queue?
- Draw a sketch of such a queue with one node.

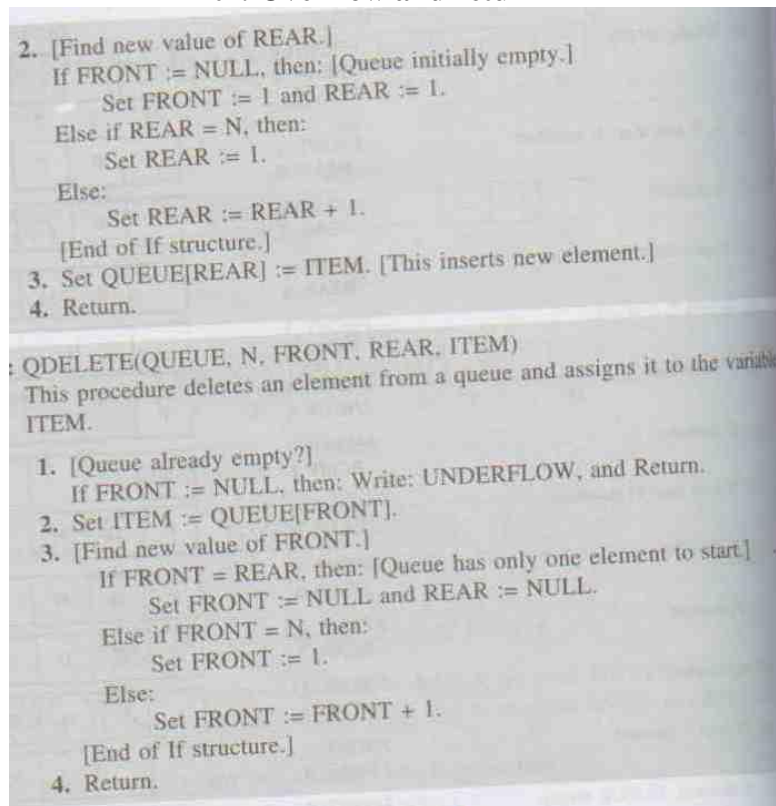
Program 14

Date : / /

Aim : Write a program to Perform the basic operation of Circular Queue using array / linked List.

Algorithm:

1. If $FRONT = 1$ and $REAR = N$ or if $FRONT = REAR + 1$ then :
Print : Overflow and return



Source code:

Output:

Selected Questions :

- "Circular Queues is better than linear queues". Justify the statement.

Program 15

Date : / /

Aim : Write a program to Perform the basic operation of Linked List.

Algorithm:

- INSLOC(INFO,LINK,START,AVAIL,LOC,ITEM)**
1. if AVAIL = NULL , then Write : OVERFLOW, and Exit
 2. Set NEW=AVAIL and AVAIL=LINK[AVAIL]
 3. set INFO[NEW]=ITEM
 4. if LOC=NULL then
 set LINK[NEW]=START and START=NEW
 else
 set LINK[NEW]=LINK[LOC] and LINK[LOC]=NEW
 5. exit.

DEL(INFO,LINK,START,AVAIL,LOC,LOCP)

1. if LOCP = NULL , then
 set START=LINK[START]
 else
 set LINK[LOCP]=LINK[LOC]
2. Set LINK[LOC]=AVAIL and AVAIL=LOC
3. exit.

Source code:

Output:

Selected Questions :

- How to implement the list operations?
- Define linked list.
- What is a traversal?
- When is a link list better than arrays
- Mention any 2 application of linked list
- Why is linked list used for polynomial arithmetic?
- When is a link list better than arrays.

Program 16

Date : / /

Aim : Write a program to Perform the basic operation of circular linked List.

Algorithm:

1. set PTR=LINK[START]
2. Repeat steps3 and 4 while PTR!=START
3. Apply PROCESS to INFO[PTR]
4. Set PTR=LINK[PTR]
5. exit

Source code:

Output:

Selected Questions :

1. Differentiate singly linked list and circularly linked list.

Program 17

Date : / /

Aim : Write a program to Perform the basic operation of doubly linked List.

Algorithm:

- INSTWL(INFO,FORW,BACK,START,AVAIL,LOCA,LOCB)**
1. if AVAIL = NULL , then Write : OVERFLOW, and Exit
 2. Set NEW=AVAIL and AVAIL=FORW[AVAIL], INFO[NEW]=ITEM
 3. set FORW[LOCA]=NEW, FORW[NEW]=LOCB
BACK[LOCB]=NEW, BACK[NEW]=LOCA
 4. exit.

Source code:

Output:

Selected Questions :

- What are the benefits of doubly linked list over singly linked list?
- What are the benefits of doubly linked list over singly linked list?
- Explain the application of doubly linked list.
- Explain in detail about insertion and deletion of elements in the doubly linked list

Program 18

Date : / /

Aim : Write a program to traverse binary tree..

Algorithm:

PREORD(INFO, LEFT, RIGHT, ROOT)

A binary tree T is in memory. The algorithm does a preorder traversal of T, applying an operation PROCESS to each of its nodes. An array STACK is used to temporarily hold the addresses of nodes.

1. [Initially push NULL onto STACK, and initialize PTR.]
Set TOP := 1, STACK[1] := NULL and PTR := ROOT.
2. Repeat Steps 3 to 5 while PTR ≠ NULL:
3. Apply PROCESS to INFO[PTR].
4. [Right child?]
If RIGHT[PTR] ≠ NULL, then: [Push on STACK.]
Set TOP := TOP + 1, and STACK[TOP] := RIGHT[PTR].
[End of If structure.]
5. [Left child?]
If LEFT[PTR] ≠ NULL, then:
Set PTR := LEFT[PTR].
Else: [Pop from STACK.]
Set PTR := STACK[TOP] and TOP := TOP - 1.
[End of If structure.]
[End of Step 2 loop.]
6. Exit.

Source code:

Output:

Selected Questions :

- Give the linked memory representation of a binary tree.
- What are the various tree traversal methods? Explain.
- List out the application of binary tree and explain
- A given binary search tree is to be sorted. Which traversal method should be applied?
- How many nodes are there on level I of a binary tree? Prove the answer.
- How do you insert an element in a binary tree
- Using recursive procedure, explain the operation of post order and pre order traversal of binary tree.
- Define terminal node, degree of a node, sibling, depth of a node

Program 19

Date : / /

Aim : Write a program to Perform the concept of recursion.

Algorithm:

FACTORIAL(FACT,N)

1. If $N=0$, then : Set $FACT = 1$ and return
2. Call $FACTORIAL(FACT,N-1)$
3. Set $FACT = N * FACT$
4. Return

Source code:

Output:

Selected Questions :

- Explain the concept of recursion
- What is iteration? Give example.
- What is meant by recursive calls? Discuss its advantages and disadvantages
- How does recursion differ from iteration?

Program 20

Date : / /

Aim : Write a program to sort the list using Heap sort.

Algorithm:

INSHEAP(TREE,N,ITEM)

1. Set $N=N+1$ and $PTR=N$
2. Repeat Steps 3 to 6 while $PTR < 1$
3. Set $PAR = \text{floor}(PTR/2)$
4. if $ITEM \leq TREE[PAR]$, then
set $TREE[PTR]=ITEM$, and return
5. Set $TREE[PTR] = TREE[PAR]$
6. SET $PTR=PAR$
7. SET $TREE[I]=ITEM$
8. RETURN

DELHEAP(TREE,N,ITEM)

1. Set $ITEM=TREE[1]$
2. Set $LAST=TREE[N]$ and $N=N-1$
3. Set $PTR=1$, $LEFT=2$ and $RIGHT=3$
4. Repeat Steps 5 to 7 while $RIGHT \leq N$
5. If $LAST \geq TREE[LEFT]$ and $LAST \geq TREE[RIGHT]$, then
Set $TREE[PTR]=LAST$ and Return.
6. IF $TREE[RIGHT] \leq TREE[LEFT]$, then
Set $TREE[PTR]=TREE[LEFT]$ and $PTR=LEFT$
Else
Set $TREE[PTR]=TREE[RIGHT]$ and $PTR=RIGHT$
7. Set $LEFT=2*PTR$ and $RIGHT=LEFT+1$
8. If $LEFT=N$ and if $LAST < TREE[LEFT]$ then Set $PTR=LEFT$
9. Set $TREE[PTR]=LAST$
10. Return

Source code:

Output:

Complexity:

Algorithm	Worst case	Average Case
Heap sort	$O(n \log n)$	$O(n \log n)$

Selected Questions :

- Define heap sort.
- What is the use of sentinel value in binary heap?