Assignment No. 9

Searching and Sorting

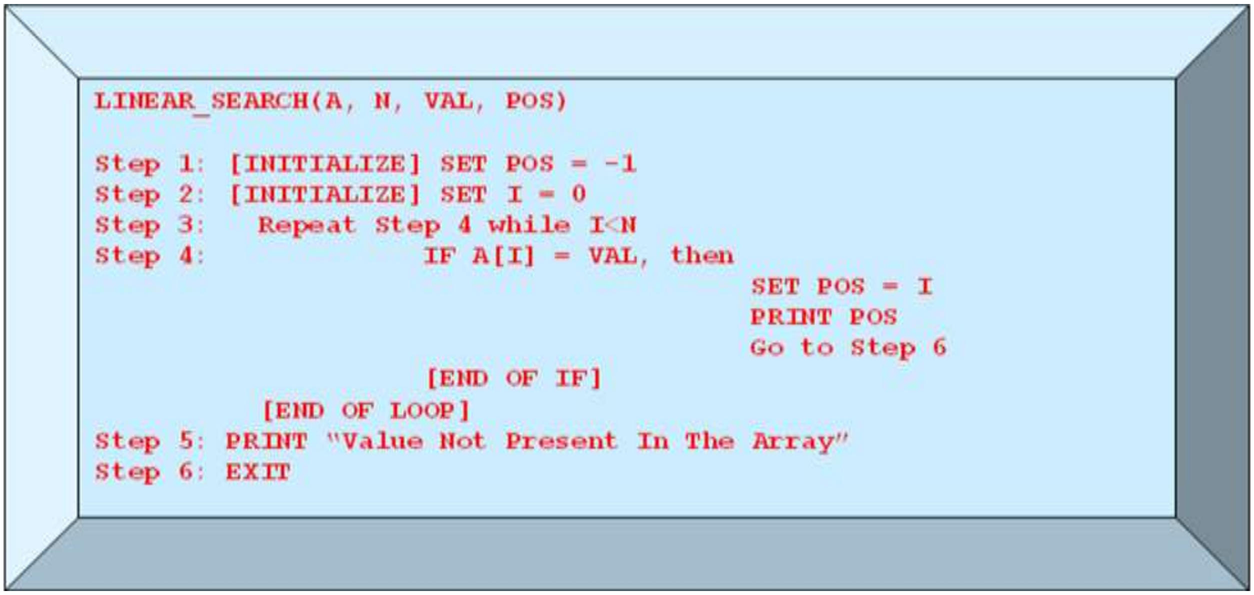
Aim
To implement the following Searching and Sorting methods: Searching: Sequential/Linear Search and Binary Search. Sorting: Bubble Sort, Insertion Sort, Selection Sort, Merge Sort, Heap Sort and Quick Sort.

Objective(s)	
1	To study searching strategies.
2	To study sorting techniques.
3	To implement searching and sorting methods.

Theory
1. State and explain: sequential search and binary search. Write detailed algorithm for the same. 2. State and explain: Bubble Sort, Insertion Sort, Selection Sort, Merge Sort, Heap Sort and Quick Sort. Write the detailed algorithm for all sorting methods.

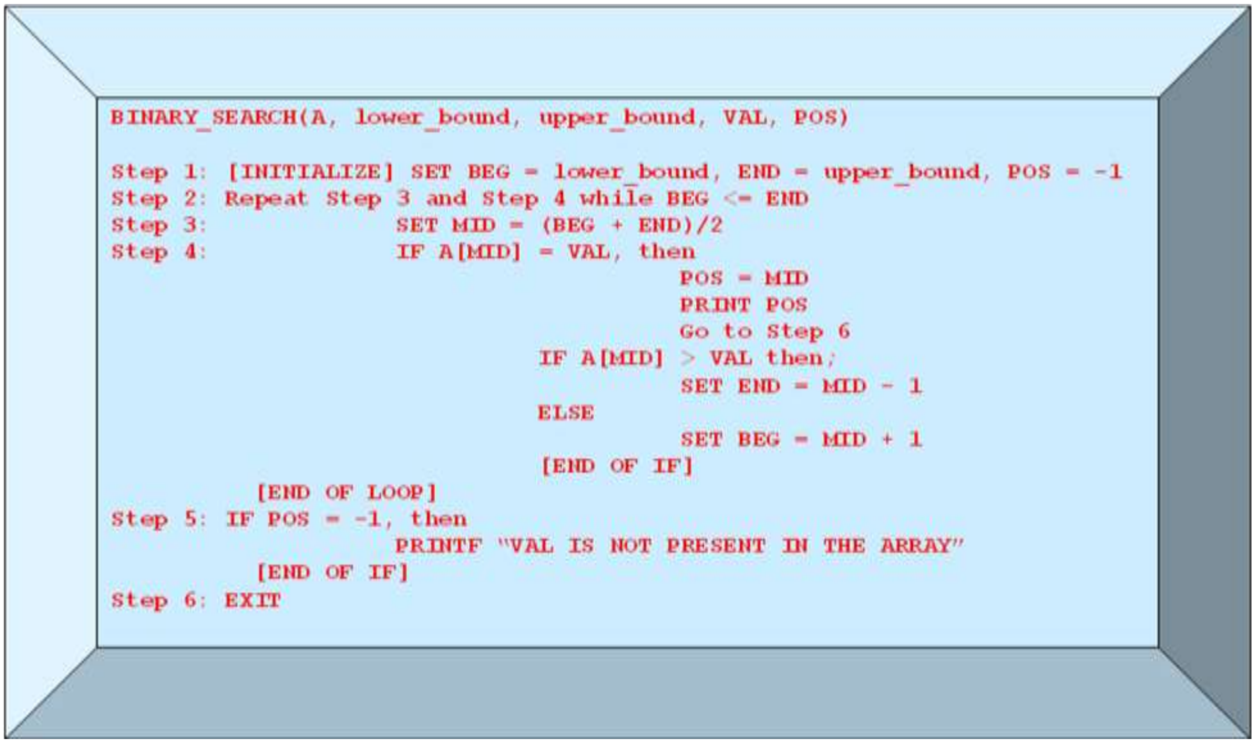
Algorithms:

Linear Search:



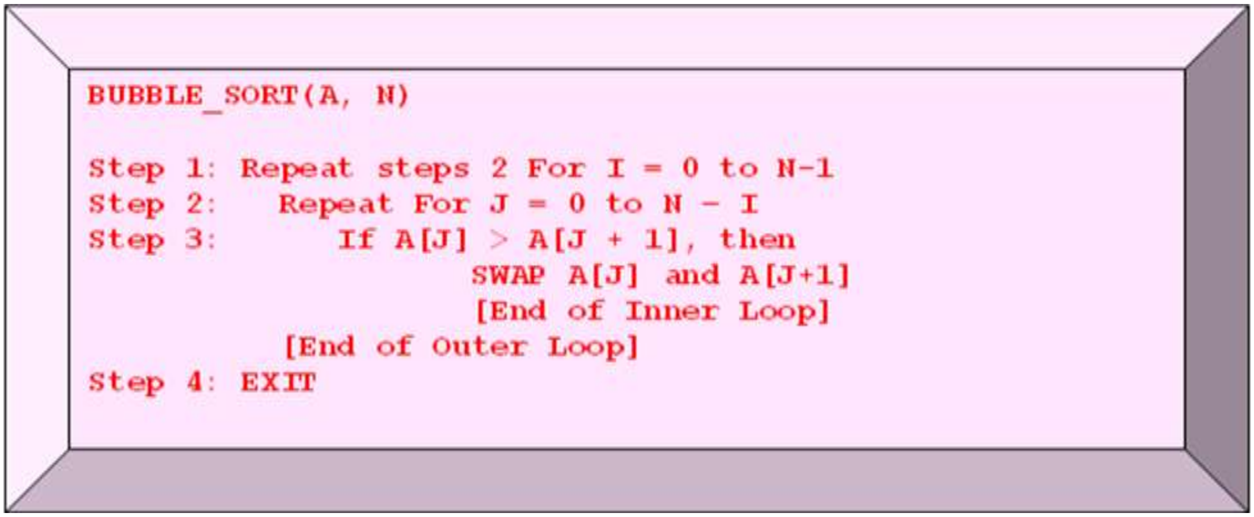
```
LINEAR_SEARCH(A, N, VAL, POS)

Step 1: [INITIALIZE] SET POS = -1
Step 2: [INITIALIZE] SET I = 0
Step 3:   Repeat Step 4 while I<N
Step 4:             IF A[I] = VAL, then
                                  SET POS = I
                                  PRINT POS
                                  Go to Step 6
                    [END OF IF]
          [END OF LOOP]
Step 5: PRINT "Value Not Present In The Array"
Step 6: EXIT
```

Binary Search:



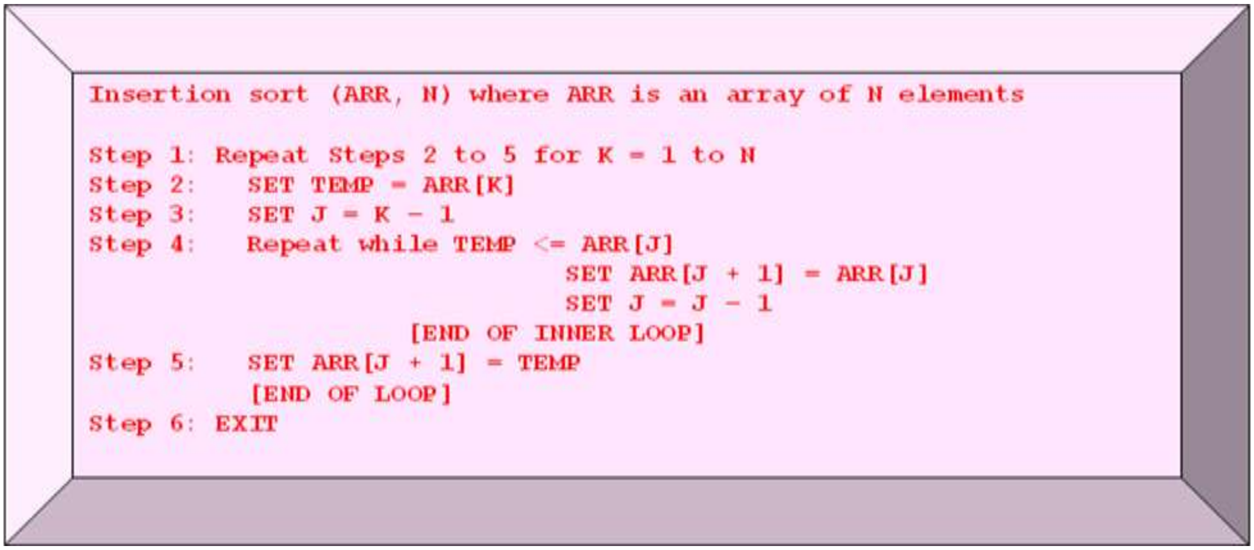
```
BINARY_SEARCH(A, lower_bound, upper_bound, VAL, POS)

Step 1: [INITIALIZE] SET BEG = lower_bound, END = upper_bound, POS = -1
Step 2: Repeat Step 3 and Step 4 while BEG <= END
Step 3:             SET MID = (BEG + END)/2
Step 4:             IF A[MID] = VAL, then
                                        POS = MID
                                        PRINT POS
                                        Go to Step 6
                              IF A[MID] > VAL then;
                                        SET END = MID - 1
                              ELSE
                                        SET BEG = MID + 1
                              [END OF IF]
          [END OF LOOP]
Step 5: IF POS = -1, then
                    PRINTF "VAL IS NOT PRESENT IN THE ARRAY"
          [END OF IF]
Step 6: EXIT
```

Bubble Sort:



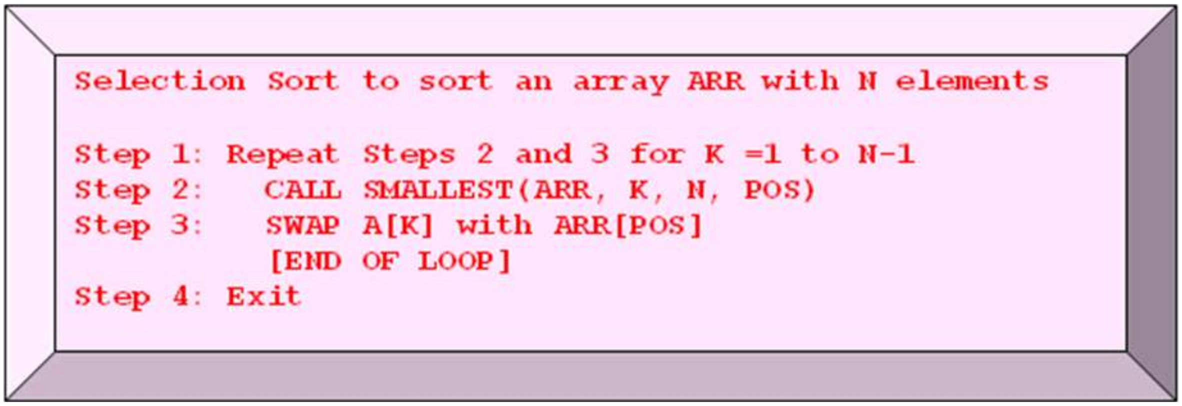
```
BUBBLE_SORT(A, N)

Step 1: Repeat steps 2 For I = 0 to N-1
Step 2:   Repeat For J = 0 to N - I
Step 3:      If A[J] > A[J + 1], then
                    SWAP A[J] and A[J+1]
                    [End of Inner Loop]
          [End of Outer Loop]
Step 4: EXIT
```

Insertion Sort:

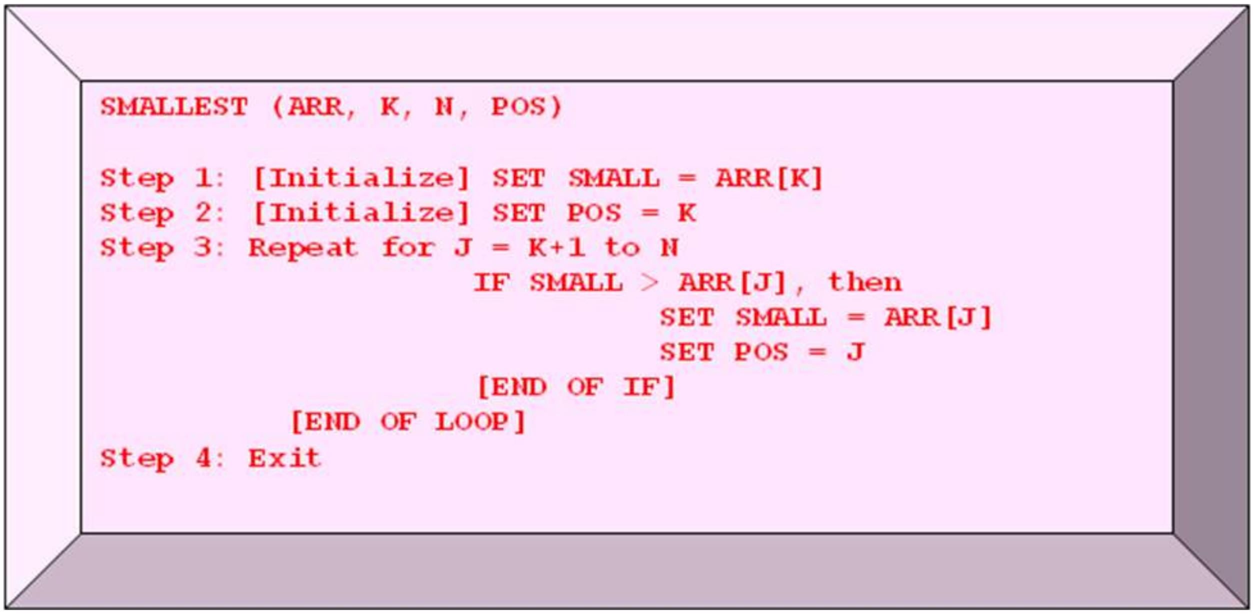


```
Insertion sort (ARR, N) where ARR is an array of N elements

Step 1: Repeat Steps 2 to 5 for K = 1 to N
Step 2:    SET TEMP = ARR[K]
Step 3:    SET J = K - 1
Step 4:    Repeat while TEMP <= ARR[J]
                                 SET ARR[J + 1] = ARR[J]
                                 SET J = J - 1
                     [END OF INNER LOOP]
Step 5:    SET ARR[J + 1] = TEMP
           [END OF LOOP]
Step 6: EXIT
```

Selection Sort:

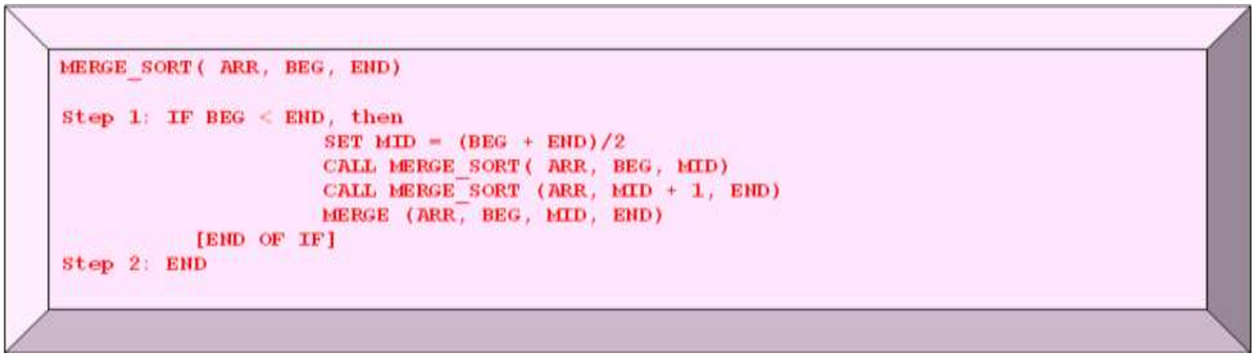


```
Selection Sort to sort an array ARR with N elements

Step 1: Repeat Steps 2 and 3 for K =1 to N-1
Step 2:   CALL SMALLEST(ARR, K, N, POS)
Step 3:   SWAP A[K] with ARR[POS]
          [END OF LOOP]
Step 4: Exit
```

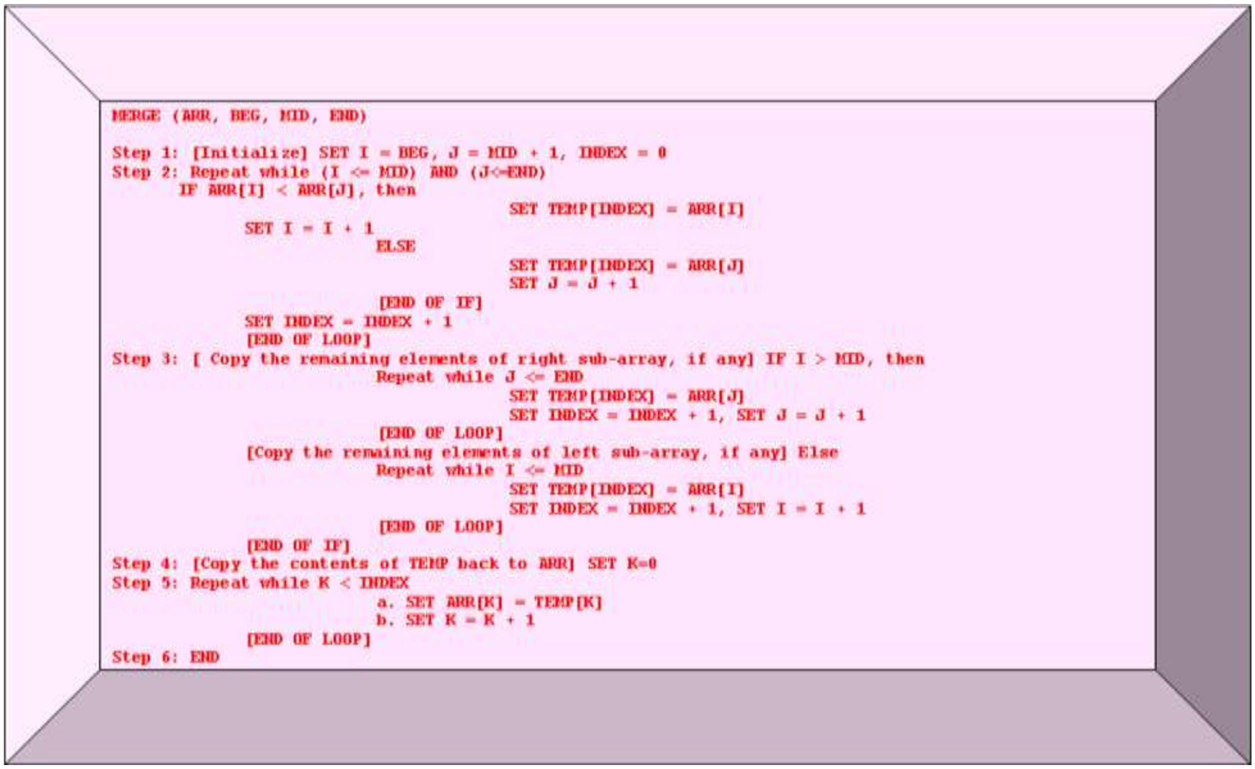


```
SMALLEST (ARR, K, N, POS)

Step 1: [Initialize] SET SMALL = ARR[K]
Step 2: [Initialize] SET POS = K
Step 3: Repeat for J = K+1 to N
                    IF SMALL > ARR[J], then
                                 SET SMALL = ARR[J]
                                 SET POS = J
                    [END OF IF]
           [END OF LOOP]
Step 4: Exit
```

Merge Sort:

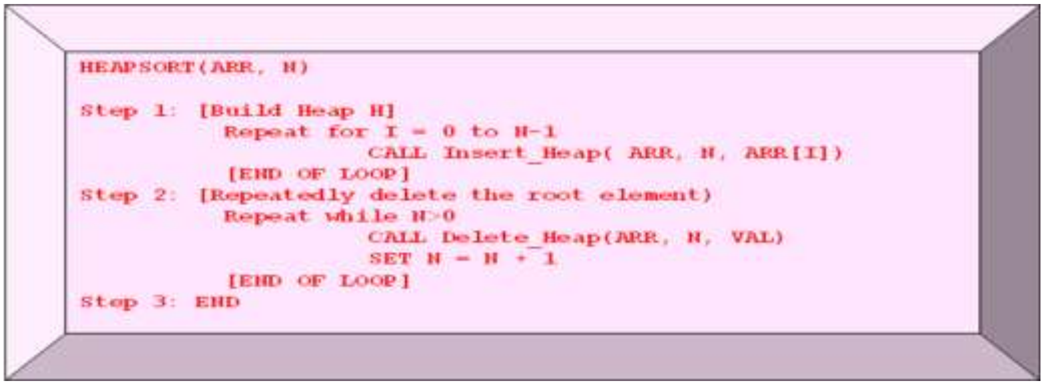


```
MERGE_SORT( ARR, BEG, END)

Step 1: IF BEG < END, then
                    SET MID = (BEG + END)/2
                    CALL MERGE_SORT( ARR, BEG, MID)
                    CALL MERGE_SORT (ARR, MID + 1, END)
                    MERGE (ARR, BEG, MID, END)
          [END OF IF]
Step 2: END
```



```
MERGE (ARR, BEG, MID, END)

Step 1: [Initialize] SET I = BEG, J = MID + 1, INDEX = 0
Step 2: Repeat while (I <= MID) AND (J<=END)
       IF ARR[I] < ARR[J], then
                                         SET TEMP[INDEX] = ARR[I]
              SET I = I + 1
                           ELSE
                                         SET TEMP[INDEX] = ARR[J]
                                         SET J = J + 1
                           [END OF IF]
              SET INDEX = INDEX + 1
              [END OF LOOP]
Step 3: [ Copy the remaining elements of right sub-array, if any] IF I > MID, then
                           Repeat while J <= END
                                         SET TEMP[INDEX] = ARR[J]
                                         SET INDEX = INDEX + 1, SET J = J + 1
                           [END OF LOOP]
              [Copy the remaining elements of left sub-array, if any] Else
                           Repeat while I <= MID
                                         SET TEMP[INDEX] = ARR[I]
                                         SET INDEX = INDEX + 1, SET I = I + 1
                           [END OF LOOP]
              [END OF IF]
Step 4: [Copy the contents of TEMP back to ARR] SET K=0
Step 5: Repeat while K < INDEX
                           a. SET ARR[K] = TEMP[K]
                           b. SET K = K + 1
              [END OF LOOP]
Step 6: END
```

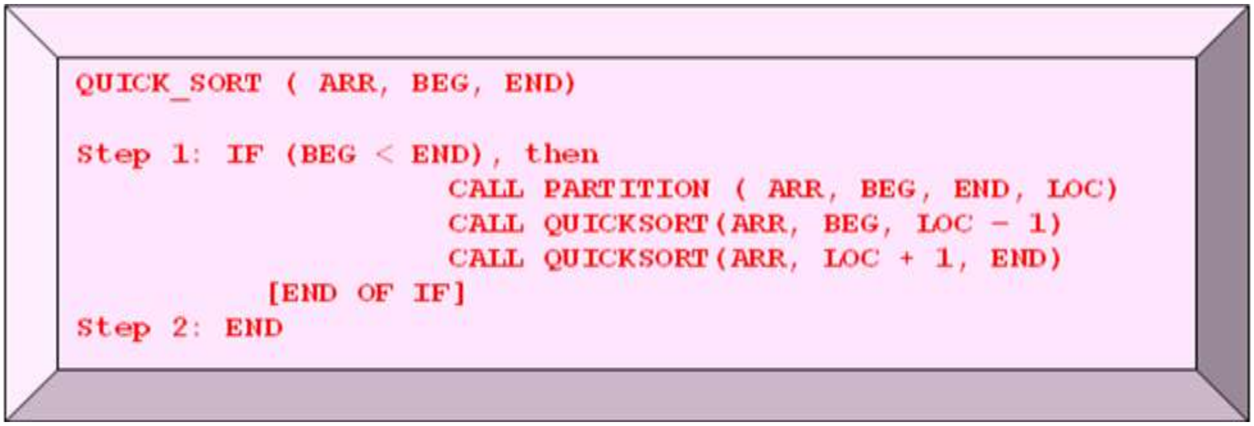
Heap Sort:



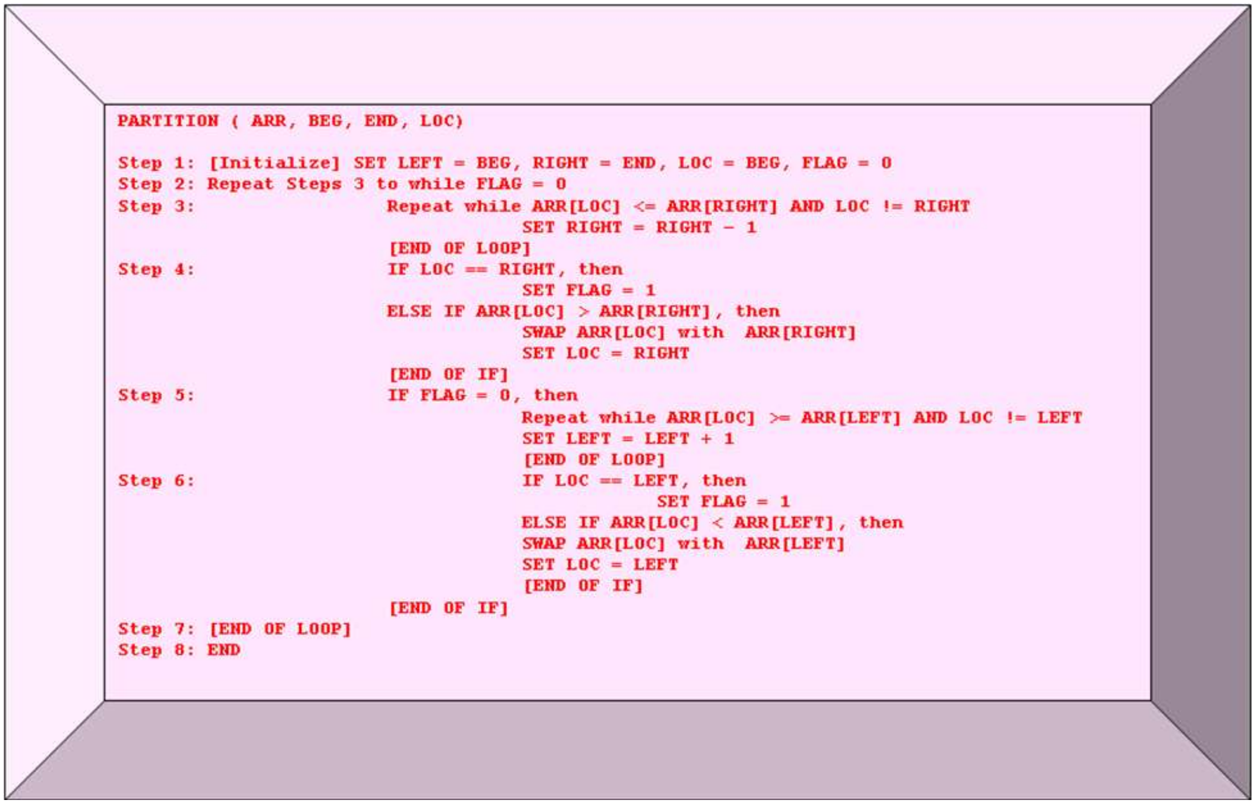
```
HEAPSORT(ARR, N)

Step 1: [Build Heap H]
          Repeat for I = 0 to N-1
                    CALL Insert_Heap( ARR, N, ARR[I])
          [END OF LOOP]
Step 2: [Repeatedly delete the root element)
          Repeat while N>0
                    CALL Delete_Heap(ARR, N, VAL)
                    SET N = N + 1
          [END OF LOOP]
Step 3: END
```

(Students should write Insert_Heap and Delete_Heap functions)

Quick Sort:



```
QUICK_SORT ( ARR, BEG, END)

Step 1: IF (BEG < END), then
                    CALL PARTITION ( ARR, BEG, END, LOC)
                    CALL QUICKSORT(ARR, BEG, LOC - 1)
                    CALL QUICKSORT(ARR, LOC + 1, END)
          [END OF IF]
Step 2: END
```



```
PARTITION ( ARR, BEG, END, LOC)

Step 1: [Initialize] SET LEFT = BEG, RIGHT = END, LOC = BEG, FLAG = 0
Step 2: Repeat Steps 3 to while FLAG = 0
Step 3:                 Repeat while ARR[LOC] <= ARR[RIGHT] AND LOC != RIGHT
                                    SET RIGHT = RIGHT - 1
                        [END OF LOOP]
Step 4:                 IF LOC == RIGHT, then
                                    SET FLAG = 1
                        ELSE IF ARR[LOC] > ARR[RIGHT], then
                                    SWAP ARR[LOC] with  ARR[RIGHT]
                                    SET LOC = RIGHT
                        [END OF IF]
Step 5:                 IF FLAG = 0, then
                                    Repeat while ARR[LOC] >= ARR[LEFT] AND LOC != LEFT
                                    SET LEFT = LEFT + 1
                                    [END OF LOOP]
Step 6:                             IF LOC == LEFT, then
                                                SET FLAG = 1
                                    ELSE IF ARR[LOC] < ARR[LEFT], then
                                    SWAP ARR[LOC] with  ARR[LEFT]
                                    SET LOC = LEFT
                                    [END OF IF]
                        [END OF IF]
Step 7: [END OF LOOP]
Step 8: END
```

Conclusion