DEPARTMENT OF COMPUTER SCIENCE
INSTITUTE OF MANAGEMENT AND RESEARCH, JALGAON

Name Warke Purva Dilip

Expt. Title write a program for creating max-heap using INSERT and ADJUST/HEAPIFY

Class SYMCA Batch _____ Performed on _____

Roll No. _____ Expt. No. _____ Submitted on _____

Remarks _____ Returned on _____

write a algorithm for creating max-heap using INSERT.

```
Procedure INSERT_MAX(A,n)
    Description :- This procedure rearranges
    elements such that maximum element is
    should be at the root or at A(1). where
    A(1:n) is array and 'n' is the number
    of element in array.

    Declaration :- integer A(1:n)
                   integer i, j, n

        j ← n ; i ← ⌊n/2⌋ ; item ← A(n)
        while i>0 and A(i) < item, do
        |   A(j) ← A(i)
        |   j ← i
        |   i ← i/2
        repeat
        A(j) ← item
end INSERT_MAX

for i ← 2 to n do
|   call INSERT_MAX
repeat.
```

Incomplete for :
1) Algorithm
2) Flow Chart
3) Programme Listing
4) Results
5) Comments

write Algorithm for creating max-heap using ADJUST_HEAPIFY

```
Procedure HEAPIFY_MAX(A,n)
    Decription :- Readjust the element in A(1:n)
                  'n' is the large non-leaf node
    Declaration :- integer n,i
        for i ← ⌊n/2⌋ to 1 by -1 do
        | call ADJUST_MAX(A,i,n)
        repeat.
end HEAPIFY_MAX

Procedure ADJUST_MAX(A,i,n)
    Description :- The complete binary trees with roots
    A(2*i) and A(2*i+1) are combine with A to from
    single heap 1≤i≤n no node has address greater
    than n.
    Declaration :- integer i,j,n
        j ← 2*i , item ← A(i)
        while j≤n do
        |   if i<n and A(j)<A(j+1), then
        |   | j ← j+1
        |   endif
        |   if item > A(j)
        |   | then exit loop
        |   else
        |   | A(⌊j/2⌋) ← A(j)
        |   | j ← 2*j
        |   endif
        repeat
        A(⌊j/2⌋) ← item
end ADJUST_MAX
```

DEPARTMENT OF COMPUTER SCIENCE
INSTITUTE OF MANAGEMENT AND RESEARCH, JALGAON

Name Warke Purva Dilip

Expt. Title Write a program for creating min heap using INSERT and AJUST / HEAPIFY

Class SYMCA Batch _____ Performed on _____

Roll No. _____ Expt. No. _____ Submitted on _____

Remarks _____ Returned on _____

Write a algorithm for creating min-heap using insert algorithm.

Procedure INSERT_MIN(A,n)

Description :- This procedure rearranges elements A(1:n) such that minimum element is should be at the root or at A(1). where n is the number of elements in array.

Declaration :- integer A(1:n)
integer i,j,n.

```
j ← n ; i ← ⌊n/2⌋ ; item ← A(n)
while i > 0 and A(i) > item, do
    A(j) ← A(i)
    j ← i
    i ← i/2
repeat
A(j) ← item
END INSERT_MIN

for i ← 2 to n, do
    call INSERT_MIN(A,i)
repeat
```

Incomplete for :
1) Algorithm
2) Flow Chart
3) Programme Listing
4) Results
5) Comments

Write an algorithm for creating min heap using ADJUST_HEAPIFY Algorithm.

Procedure ~~ADJUS~~ HEAPIFY_MIN(A,n)

Description :-
This procedure readjust the elements in A such that to form an heap(min). where n is number of elements and A(1:n) is an array.

Declaration :- integer A,n,i

```
for i ← ⌊n/2⌋ to 1 by -1, do
  | call ADJUST_MIN (A,i,n)
repeat
end HEAPIFY_MIN
```

procedure ADJUST_MIN (A,i,n)

Description :-
This procedure ADJUST_MIN the nodes in tree whose root is at location i. n is the number of elements in an array of A(1:n)

Declaration :- integer A,n,
integer i,j, item

```
j ← 2*i , item ← A(i)
while j≤n , do
  | if j<n and A(j)>A(j+1)
  |   | j ← j+1
  | endif
  | if item < A(j), then
  |   | exit loop
  | else
  |   | A(⌊j/2⌋) ← A(j)
  |   | j ← j*2
  | endif
repeat
A(⌊j/2⌋) ← item
end ADJUST_MIN
```

DEPARTMENT OF COMPUTER SCIENCE
INSTITUTE OF MANAGEMENT AND RESEARCH, JALGAON

Name Warke Purva Dilip
Expt. Title Write a program for creating Heap Sort (Ascending)
Class SYMCA Batch _____ Performed on _____
Roll No. _____ Expt. No. _____ Submitted on _____
Remarks _____ Returned on _____

write a algorithm to sort an array .in ascending order using Heap Sort.

procedure HEAP_SORT(A,n)
Description :-
This procedure sorts the n elements of A(1:n). Heap Sort rearrange them in-place into non-decreasing order. where A(1:n) is array contains 'n' number of elements.

Declaration :- integer A,n
integer i.

```
// Transforms the elements into a heap
   call HEAPIFY(A,n)
// interchange the maximum with the elements
   at the end n and adjust root.
   for i ← n to 2 by -1 do
      | call EXCHANGE(A(1),A(i))
      | Call ADJUST(A,1,i-1)
   repeat
end HEAP_SORT
```

Procedure HEAPIFY(A,n)
Description :-
This procedure readjust the elements in A(1:n) such to form an heap (max)

Declaration :- integer n, i.

Incomplete for :
1) Algorithm
2) Flow Chart
3) Programme Listing
4) Results
5) Comments

```
        for i ← ⌊n/2⌋ to 1 by -1, do
        |   call ADJUST(A,n)
        repeat
end  HEAPIFY
```

```
Procedure ADJUST (A,i,n)
```

Description :-

This procedure is for binary tree whose root is at location i. n is the number of elements is an array A(1:n)

Declaration :- integer i, j, item.

```
    j ← 2*i
    item ← A(i)
    while j≤n, do
    |   if j<n and A(j)<A(j+1), then
    |   |   j ← j+1
    |   endif
    |   if item > A(j), then
    |   |   exit loop
    |   else
    |   |   A(⌊j/2⌋) ← A(j)
    |   |   j ← j*2
    |   endif
    repeat
    A(⌊j/2⌋) ← item
end  ADJUST
```

DEPARTMENT OF COMPUTER SCIENCE
INSTITUTE OF MANAGEMENT AND RESEARCH, JALGAON

Name Wake Purva Dilip

Expt. Title Write a program for creating Heap-sort (Descending)

Class SYMCA Batch _____ Performed on _____

Roll No. _____ Expt. No. _____ Submitted on _____

Remarks _____ Returned on _____

Write a algorithm to sort an array in descending order using Heap sort.

```
Procedure HEAP_SORT (A,n)
  Description :-
      This procedure sorts the n elements
  of A(1:n). heap sort rearrange then
  in-place into decreasing order. where A(1:n)
  is array contain 'n' number of elements.
  Declaration :- integer A,n
                 integer i.

   // Transform the elements into a heap
    call HEAPIFY(A,n)
    // interchange the minimum with the elements
      at the end n and adjust root.
    for i←n to 2 by -1.do
      |  call EXCHANGE (A(1), A(i))
      |  call ADJUST(A, 1, i-1)
    repeat
end HEAP_SORT

procedure HEAPIFY(A,n)
  Description :—
      This procedure readjust the elements
  in A(1:n) such to form an heap (min)

Declaration :— integer n, i, A
```

Incomplete for :
1) Algorithm
2) Flow Chart
3) Programme Listing
4) Results
5) Comments

```
    for i ← ⌊n/2⌋ to 1 by -1, do
    |   Call ADJUST (A, i, n)
    repeat
end HEAPIFY
```

Procedure ADJUST (A, i, n)

Description :— This procedure ADJUST the nodes in tree whose root is at location i. 'n' is the number of elements in an array of A(1:n)

Declaration :- integer A, n.
integer i, j, item

```
j ← 2*i , item ← A(i)
while j≤n . do
|   if j<n and A(j) > A(j+1)
|   |   j ← j+1
|   endif
|   if item < A(j) , then
|   |   exit loop
|   else
|   |   A(⌊j/2⌋) ← A(j)
|   |   j ← j*2
|   endif
repeat
A(⌊j/2⌋) ← item
```

end ADJUST.

DEPARTMENT OF COMPUTER SCIENCE
INSTITUTE OF MANAGEMENT AND RESEARCH, JALGAON

Name Warke Purva Dilip

Expt. Title write a program to find solution of knapsack instant

Class SYMCA Batch _____ Performed on _____

Roll No. _____ Expt. No. _____ Submitted on _____

Remarks _____ Returned on _____

write a algorithm for greedy knapsack method.

Procedure GREEDY_KNAPSACK(P, W, M, X, n)

assumming the data given is sorted according to P/W ratio.

Description :- P(1:n) and W(1:n) contain the profits and weights resp. of nth objects ordered so that $P(i)/W(i) \geq P(i+1)/W(i+1)$. M is the max of size and X(1:n) is the solution vector.

Declaration :- real P(1:n), W(1:n), X(1:n), M, Cu
integer i, n.

```
X ← 0
Cu ← M
for i ← i to n do
    if W(i) > Cu, then
        exit loop
    else
        X(i) ← 1
        Cu ← Cu - W(i)
    endif
repeat
if i ≤ n, then
    X(i) ← Cu / W(i)
endif
```

end GREEDY_KNAPSACK.

Incomplete for :
1) Algorithm
2) Flow Chart
3) Programme Listing
4) Results
5) Comments

DEPARTMENT OF COMPUTER SCIENCE
INSTITUTE OF MANAGEMENT AND RESEARCH, JALGAON

Name Warke Purva Dilip

Expt. Title Write a program to find minimum-cost Spanning Trees.

Class SYMCA Batch _____ Performed on _____

Roll No. _____ Expt. No. _____ Submitted on _____

Remarks _____ Returned on _____

Write a Algorithm to find minimum-cost Spanning Trees (Prim's & Kruskal's Algorithm)

1) Prim's Algorithm:-

Procedure PRIM (E, COST, n, T, mincost)

Description :- E is the set of edges in G. COST(n,n) is Cost adjacency matrix of an n vertex graph such that COST(i,j) is either a positive real number or $+\infty$ if no edge (i,j) exists. A mininum spanning tree is computed and stored as a set of edges in the array T(1:n,2), T(i,1), T(i,2) is an edge in the minimum cost spanning tree. The final cost is assign to min cost.

Declaration :- real COST(n,n), mincost
integer NEAR(n), n, i, j, k, l,
T(1:n-1, 2).

```
(k, l) ← edge with mincost.
mincost ← COST(k, l)
(T(1,1), T(1,2)) ← (k, l)
// fill up near array.
for i ← 1 to n do
    if COST(i, l) < COST(i, k)
        NEAR(i) ← l
    else
        NEAR(i) ← k
    endif
repeat
```

Incomplete for :
1) Algorithm
2) Flow Chart
3) Programme Listing
4) Results
5) Comments

```
NEAR(k) ← 0
NEAR(l) ← 0

// find out remaining (n-2) edges of the tree

for i ← 2 to n-1 do
    Let j be an index such that NEAR(j) ≠ 0
    and COST(j, NEAR(j)) is minimum.
    (T(i,1), T(i,2)) ← (j, NEAR(j))
    mincost ← mincost + COST(j, NEAR(j))
    NEAR(j) ← 0

    //update NEAR array
    for k ← 1 to n do
        if NEAR(k) ≠ and COST(k, NEAR(k)) >
                                (COST(k, j)
            NEAR(k) ← j
        endif
    repeat
repeat
if mincost ≥ ∞, then
    print ("NO spanning Tree")
endif
end PRIM
```

DEPARTMENT OF COMPUTER SCIENCE
INSTITUTE OF MANAGEMENT AND RESEARCH, JALGAON

Name Warke Purva Dilip

Expt. Title Write a program to implement union & find operation

Class SYMCA Batch _____ Performed on _____

Roll No. _____ Expt. No. _____ Submitted on _____

Remarks _____ Returned on _____

Procedure U(i, j)

Description :- Replace the disjoint sets with roots i, j, $i \neq j$ by their union.

Declaration :- integer i, j

PARENT(i) ← j

end U

Procedure F(i)

Description :- find the root of the tree containing ele i.

Declaration :- integer i, j.

```
j ← i
while PARENT(j) > 0
      j ← PARENT
repeat
return(j)
```

end F

Procedure UNION(i, j)

Description :- UNION sets with roots i and j, $i \neq j$, using the weighting rule. PARENT(i) = -COUNT(i) & PARENT(i) = -COUNT(j)

Declaration :- integer i, j, x

Incomplete for :
1) Algorithm
2) Flow Chart
3) Programme Listing
4) Results
5) Comments

```
        x ← PARENT(i) + PARENT(j)
        if (PARENT(i) > PARENT(j)) then
              PARENT(i) ← j ;
              PARENT(j) ← x;
        else
              PARENT(j) ← i ;
              PARENT(i) ← x;
        endif
end UNION
```

```
Procedure FIND(i)
```

Description :- Find the root of the tree containing element i, use the collapsing rule to collapse all nodes from i to the root.

```
Declaration :- integer i, j, k

    j ← i    //find first root of tree
    while PARENT(j) > 0
          j ← PARENT(j);
    repeat
    k ← i
    while k ≠ j do  //collapse nodes from k to root j
          temp ← PARENT(k);
          PARENT(k) ← j ;
          k ← temp;
    repeat
    return(j)
end FIND
```

DEPARTMENT OF COMPUTER SCIENCE
INSTITUTE OF MANAGEMENT AND RESEARCH, JALGAON

Name Warke Purva Dilip
Expt. Title write a program for merge sort.
Class SYMCA Batch _____ Performed on _____
Roll No. _____ Expt. No. _____ Submitted on _____
Remarks _____ Returned on _____

Write an Algorithm for merge sort.

```
Procedure  MERGE_SORT (low, high)
    Description :- A[low : high] is a global array
                   to be sorted. In this case the
                   list is already sorted.
    Declaration :- integer A, low, high

    if low < high, then
        mid ← (low + high)/2
        call MERGE_SORT (A, low, mid)
        call MERGE_SORT (A, mid+1, high)
        Call MERGE (A, low, mid, high)
    end if
end MERGE_SORT
```

```
Procedure MERGE (A, low, mid, high)
    Description :- This process merge two sublist
                   A(low : mid) & B(mid+1 : high) it
                   uses auxiliary B(low : high) & sorted.

    Declaration :- Global A(low : mid) & A(mid+1, high
                   integer A, low, mid, high
                   local integer i, j, k

    i ← low
    j ← mid+1
    k ← low
```

Incomplete for :
1) Algorithm
2) Flow Chart
3) Programme Listing
4) Results
5) Comments

```
    while i ≤ mid and j≤high, do
        if A(i)≤A(j), then
            B(k) ← A(i)
            i ← i+1
        else
            B(k) ← A(j)
            j ← j +1
        endif
        k ← k+1
    repeat
    if i≤mid
        while i ≤ mid, do
            B(k) ← A(i)
            i ← i+1
            k ← k+1
        repeat
    else
        while j ≤ high, do
            B(k) ← A(j)
            j ← j+1
            k ← k+1
        repeat
    endif
    for k ← low to high, do
        A(k) ← B(k)
    repeat
end MERGE
```

DEPARTMENT OF COMPUTER SCIENCE
INSTITUTE OF MANAGEMENT AND RESEARCH, JALGAON

Name Warke Purva Dilip

Expt. Title Write a program to find Minimum - Cost Spanning Tree.

Class SYMCA Batch _____ Performed on _____

Roll No. _____ Expt. No. _____ Submitted on _____

Remarks _____ Returned on _____

Procedure KRUSKAL(E, cost, n, t, mincost)

Description :- E is the set of edges in G. G has n vertices. COST(u,v) is the COST of edge COST(u,v). t is the set of edges in the minimum spanng tree. & mincost is COST.

Declaration :- real mincost, COST(1:n, 1:n)
integer PARENT(1:n),
T(1:n-1, 2), n.

Construct a heap out of the edge costs using Heapify;
for i ← 1 to n do PARENT(i) ← -1;
i ← 0 mincost ← 0;
while i < n-1 and HEAP is not empty do
Delete a minimum cost edge (u,v) from the heap and reheapify using Adjust;
j ← Find(u); & k ← find(v);
if (j ≠ k) then
i ← i+1;
t(i,1) ← t(i,2) ← v;
mincost ← mincost + cost(u,v);
UNION (j,k)
endif
repeat
if i ≠ n-1, then
print ("NO spanning Tree")
endif
end KRUSKAL.

Incomplete for :
1) Algorithm
2) Flow Chart
3) Programme Listing
4) Results
5) Comments

DEPARTMENT OF COMPUTER SCIENCE
INSTITUTE OF MANAGEMENT AND RESEARCH, JALGAON

Name Warke Purva Dilip
Expt. Title Write a program to find shortest Path using single source shortest path.
Class SYMCA Batch _____ Performed on _____
Roll No. _____ Expt. No. _____ Submitted on _____
Remarks _____ Returned on _____

Write a algorithm to find shortest Path using Single Source Shortest Path.

Procedure SHORTEST_PATH (v, COST, DIST, n)

Description :-

DIST(j), j is $1 \leq j \leq n$ is set to the length of the shortest path from vector v to vertex j in a diagraph G with n vertices. DIST(v) is set to zero. G is represented by its cost adjacency matrix COST(1:n, 1:n)

Declaration :- boolean S(1:n),
real COST(1:n, 1:n), DIST(1:n)
integer n, u, v, w, num, i.

// Initialize set S to empty & DIST using current edges.

```
for i ← 1 to n do
    S(i) ← 0;
    DIST(i) ← COST(v,i)
repeat
S(v) ← 1;
DIST(v) ← 0
for num ← 2 to n-1 do
    //choose u from among those vertices
      not in S such that
      DIST(u) = min{DIST(w) and S(w)=0}
  S(u) ← 1  // put u in S.
```

Incomplete for :
1) Algorithm
2) Flow Chart
3) Programme Listing
4) Results
5) Comments

```
        for (each w adjacent to u with S(w) = 0) do
                // update distances.
                if (DIST(w) > DIST(u) + COST(u,w)) then
                        DIST(w) ← DIST(u) + COST(u,w);
                endif
        repeat
    repeat
end SHORTEST_PATH
```

DEPARTMENT OF COMPUTER SCIENCE
INSTITUTE OF MANAGEMENT AND RESEARCH, JALGAON

Name Warke Purva Dilip

Expt. Title Write program for searching element from given array using binary search for n=1000, 2000, 3000 find exact time of execution time.

Class SYMCA Batch _____ Performed on _____

Roll No. _____ Expt. No. _____ Submitted on _____

Remarks _____ Returned on _____

BINARY SEARCH :-

Procedure BINSEARCH(A, n, x, j)

Description :- Given an Array A[1:n] of elements in nondecreasing order, $n \geq 0$, determine whether x is present and if so, return j such that $x = A[j]$; else return 0.

Declaration :- integer low, high, mid, j, n

```
low ← 1 ; high ← n
while low ≤ high , do
        mid ← [(low + high) /2]
        case
          : x < A(mid) : high ← mid -1
          : x > A (mid ) : low ← mid +1
          : else
                  j ← mid ; return

        endcase
    repeat
    j ← 0
END BINSEARCH
```

Incomplete for :

1) Algorithm
2) Flow Chart
3) Programme Listing
4) Results
5) Comments

Procedure BIN_SEARCH1 (A, n, x, j)

Description :—

Declaration :- integer low, high, mid, j, n

```
low ← 1 ; high ← n+1
while low < high -1, do
      mid ← [(low + high)/2]
      if x < A(mid), then
            high ← mid
      else
            low ← mid
      endif
repeat
if x = A(low), then
      j ← low
else
      j ← 0
endif
```

END BIN_SEARCH1

DEPARTMENT OF COMPUTER SCIENCE
INSTITUTE OF MANAGEMENT AND RESEARCH, JALGAON

Name Warke Purva Dilip

Expt. Title Write a program to find minimum & maximum from a given array

Class SYMCA Batch _____ Performed on _____

Roll No. _____ Expt. No. _____ Submitted on _____

Remarks _____ Returned on _____

Write an Algorithm to find Minimum and maximum from given array.

```
Procedure MAXMIN (P,q, max, min)

    Description :- A(1:n) is a global array.
                   The effect is to set max and
                   min to the largest and smallest
                   values in A[P:q], respectively.

    Declaration :- integer P,q
                   1 ≤ P ≤ q ≤ n


if p=q, then
|   max ← min ← A(P)
else
|   if p=q-1, then
|   |   if A(P) > A(q), then
|   |   |   max ← A(p)
|   |   |   min ← A(q)
|   |   else
|   |   |   max ← A(q)
|   |   |   min ← A(P)
|   |   endif
|   else
|   |   m ← (p+q)/2
|   |   Call MAXMIN (p, m, fmax, fmin)
|   |   Call MAXMIN (m+1, q, hmax, hmin)
|   |   if fmax > hmax, then
|   |   |   max ← fmax
|   |   else
|   |   |   max ← hmax
|   |   endif
```

Incomplete for :
1) Algorithm
2) Flow Chart
3) Programme Listing
4) Results
5) Comments

```
            if fmin < hmin, then
                min ← fmin
            else
                min ← hmin
            endif
        endif
    endif
end MAXMIN
```

DEPARTMENT OF COMPUTER SCIENCE
INSTITUTE OF MANAGEMENT AND RESEARCH, JALGAON

Name Warke Purva Dilip

Expt. Title Write a program to implement breadth & depth first search

Class SYMCA Batch B4 Performed on

Roll No. 136 Expt. No. Submitted on

Remarks Returned on

Incomplete for :
1) Algorithm
2) Flow Chart
3) Programme Listing
4) Results
5) Comments

Breadth first search :-

```
Procedure BFS(v)
```

Description :- A breadth first search of G is carried out beginning at vertex V. All vertices visited are marked as VISITED(i)=1. The graph G and array VISITED are global and VISITED is initialised to 0.

Declaration :-

```
VISITED (v) ← 1;
  u ← v
Initialize Q to be empty
loop
    for all vertices w adjecent from u do
        if VISITED(w)=0 , then
            call ADDQ(w;Q)
            VISITED (w) ← 1
        endif
    repeat
    if Q is empty then
        return
    end if
    call DELETEQ(u,Q)
repeat
end BFS.
```

Depth first search (Recursive) :—

Procedure DFS(v)

Description :- Given an undirected or directed graph G = (V, E) with n vertices & an array VISITED (n) initially set to 0.

Declaration :-

```
VISITED(v) ← 1
for each vertex w adjecent from v do
    if VISITED (w) = 0, then
        Call DFS(w)
    endif
repeat
end DFS
```

Depth first search (Non-Recursive) :—

Procedure NR_DFS(v)

```
VISITED (v) ← 1
u ← v
//initialise STACK to be empty
loop
    for all w adjecent from u do
        PUSH (w)
        VISITED (w) ← 1
    repeat
    if STACK u empty, then
        return
    else
        u ← POP ()
    endif
repeat
end NR-DFS
```

DEPARTMENT OF COMPUTER SCIENCE
INSTITUTE OF MANAGEMENT AND RESEARCH, JALGAON

Name Warke Purva Dilip
Expt. Title write a program to find shortest path using all pair path
Class SYMCA Batch B4 Performed on
Roll No. 136 Expt. No. Submitted on
Remarks Returned on

write an Algorithm for find shortest path using all pair path.

Procedure ALL_PATHS (COST, A, n)

Description :- COST(n,n) is the cost adjacency matrix of a graph with n vertices; A(i,j) is the cost of a shortest path from V_i to V_j. COST(i,i) = 0 , $1 \leq i \leq n$

Declaration :- integer i, j, k, n;
real COST(n,n) A(n,n)

```
for i ← 1 to n do
    for j ← 1 to n do
        A(i,j) ← COST(i,j)
    repeat
repeat
for k ← 1 to n do
    for i ← 1 to n do
        for j ← 1 to n do
            A(i,j) ← min{A(i,j),
                         A(i,k) + A(k,j)}
        repeat
    repeat
repeat
end ALL_PATHS
```

Incomplete for :
1) Algorithm
2) Flow Chart
3) Programme Listing
4) Results
5) Comments

DEPARTMENT OF COMPUTER SCIENCE
INSTITUTE OF MANAGEMENT AND RESEARCH, JALGAON

Name Warke Purva Dilip
Expt. Title Write a program to find all solutions for 8-queen problem using backtracking
Class SYMCA Batch B4 Performed on
Roll No. 136 Expt. No. Submitted on
Remarks Returned on

write an algorithm to find all solutions for 8-queen problem using backtracking.

```
Procedure NQUEENS(n)

  Description :- Using backtracking, this
                 procedure prints all posible place-
                 ments of n queens on an nxn
                 chessboard so that they are
                 nonattacking
  Declaration :- integer k, n, x(1:n)

  k ← 1 ; x(k) ← 0  // start with first rows &
                       0th column
  while k > 0 do // try all possible solutions.
      x(k) ← x(k) + 1
      while x(k) ≤ n and NOT PLACE(k) do
          x(k) ← x(k) + 1 //try next column & pos.
      repeat
      if x(k) ≤ n, then
          if k = n, then
              print(x)
          else
              k ← k+1
              x(k) ← 0
          endif
      else
          k ← k-1
      endif
  repeat
end NQUEENS
```

Incomplete for :

1) Algorithm
2) Flow Chart
3) Programme Listing
4) Results
5) Comments

Procedure PLACE(k)

Description :- return true if a queen can be place in k^{th} row and $x(k)^{th}$ column. otherwise it returns false. x is a global array whose first k values having set ABS(x) returns absulute value of x

Declaration :- global x(1:n)
integer i,k

```
for i ← 1 to k-1 do
    if (x(i) = x(k) OR
        ABS(i-k) = ABS(x(i)-x(k)), then
        return (false)
    endif
repeat
return (true)
end PLACE
```

DEPARTMENT OF COMPUTER SCIENCE
INSTITUTE OF MANAGEMENT AND RESEARCH, JALGAON

Name Warke Purva Dilip
Expt. Title write a program for Quick Sort (Ascending)
Class SYMCA Batch B4 Performed on
Roll No. 136 Expt. No. Submitted on
Remarks Returned on

write an Algorithm for Quick Sort.

```
Procedure QUICK_SORT(p,q)
    Description :- Sorts the elements a(p), ... a(q)
                   which reside in the global array
                   a(1:n) into ascending order;
                   a(n+1) is considered to be defined
                   and must be ≥ all the elements
                   in a(1:n).

    Declaration :- Integer p,q
                   Global n, A(n)

    if (p<q) then
        j ← PARTITION(a, p, q+1);
        Call QUICK_SORT(p, j-1);
        Call QUICK_SORT(j+1, q);
    endif
end QUICK_SORT
```

```
Procedure PARTITION(a,m,p)
    Description :- within a(m); a(m+1), ..., a(p-1)
                   the elements are rearranged in
                   such a manner that if initially
                   t = a(m), then after completion
                   a(q) = t for some q between m &
                   p-1, a(k) ≤ t for m ≤ k < q and
                   a(k) ≥ t. for q < k < p. q is returned
                   set a(p) = ∞
```

Incomplete for :
1) Algorithm
2) Flow Chart
3) Programme Listing
4) Results
5) Comments

```
Declaration :- Global A(m:p)
               integer m,p.

    V ← A(m);
    i ← m
    j ← p
    loop
        do
            i ← i+1
            while (A(i) ≥ V
            repeat
                j ← j-1
            while A(j ≤ V)
            if (i<j) then
                EXCHANGE (a, i, j);
            else
                exit loop
            endif
        repeat
        A(m) ← A(j)
        A(j) ← V
        return j
end PARTITION

Procedure EXCHANGE (a, i, j)
    Description:- Exchange A(i) with A(j)
        P ← A(i);
        A(i) ← A(j);
        A(j) ← P

end EXCHANGE.
```

DEPARTMENT OF COMPUTER SCIENCE
INSTITUTE OF MANAGEMENT AND RESEARCH, JALGAON

Name Warke Purva Dilip
Expt. Title Write a program to find longest common subsequence
Class SYMCA Batch B4 Performed on
Roll No. 136 Expt. No. Submitted on
Remarks Returned on

1) Procedure LCS_LENGTH(x,y)

Description :- $x = (x_1, x_2, \ldots x_m)$ & $y = (y_1, y_2, \ldots y_n)$ are the two given sequences. The algo uses two mxn matrices, c(0:m,0:n) and B(0:m,0:n) The matrix c stores the length of lcs & matrix B stores symbols S which are we to construct LCS. The entries are computed in row order. This procedure return matrix B & C

Declaration :-

```
global integer c(0:m,0:n)
char B(0:m,0:n)
char X(1:m), Y(1:n)
local integer m,n,i,j
```

```
m ← LENGTH(X)
n ← LENGTH(Y)
for i ← 0 to m do
    c(i,0) ← 0
repeat
for j ← 0 to n do
    c(0,j) ← 0
repeat
for i ← 1 to m do
    for j ← 1 to n do
        if X(i) = Y(j), then
            c(i,j) ← c(i-1,j-1)+1
            B(i,j) ← '↖'
        else
```

Incomplete for :
1) Algorithm
2) Flow Chart
3) Programme Listing
4) Results
5) Comments

```
        else
            if c(i-1,j) ≥ c(i,j-1) then
                c(i,j) ← c(i-1,j)
                B(i,j) ← '↑'
            else
                c(i,j) ← c(i,j-1)
                B(i,j) ← '←'
            endif
        endif
    repeat
repeat
return X & Y
END LCS_LENGTH
```

2) LCS Non-recursive

```
procedure LCS_PRINT(B,X,m,n)
    Declaration : Global char B(0:m,0:n)
                  global X(x1,x2,x3,---xm)
                  integer m,n
                  Local integer i,j
                  STACK S(1:k)
    for i ← m to 1 by -1 do
        for j ← n to 1 by -1 do
            if B(i,j) = '↖' then
                PUSH(X(i))
                i ← i-1
            else
                if B(i,j) = '↑' then
                    i ← j-1 and j ← j+1
                endif
            endif
        repeat
    repeat
    for i ← top to 1 do
        PRINT(STACK(i))
    repeat
END LCS_PRINT
```