DEPARTMENT OF COMPUTER SCIENCE
INSTITUTE OF MANAGEMENT AND RESEARCH, JALGAON

Name Badhe Rohan Vikas
Expt. Title write a program for creating Max-heap using INSERT & ADJUST
Class SYMCA Batch B1 Performed on
Roll No. 04 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 & '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 & 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
```

while Algorithm for creating max-heap using ADJUST HEAPIFY

Procedure HEAPIFY_MAX(A,n)
Decription :- integer n,i

```
for i ← [n/2] to 1 by -1 do
  | call ADJUST_MAX(A,i,n)
repeat
end HEAPIFY_MAX
```

Listing

Procedure ADJUST_MAX (A, i, n)
Description :- 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 tree with roots A(2*i+1) are combine with A to form single $1 \le i \le n$ no node has address greater than n.

```
Declaration :- integer i, j, n
    j ← 2*i, item ← A(i)
    while j ≤ n do
    |    if j<n & 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 Badhe Rohan Vikas

Expt. Title ~~write a program for creating min heap using~~ INSERT ADJUST / HEAPIFY

Class SYMCA Batch B1 Performed on

Roll No. 04 Expt. No. Submitted on

Remarks Good 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 in 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 & 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
```

write an algorithm for creating min heap using ADJ
HEAPIFY Algorithm.

Procedure HEAPIFY_MIN(A,n)

Description :-

This procedure readjust the elements in a suc
to form an heap (min). where n is number of elem
& 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
root is at location i, n is the number of elements
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 & 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 Badhe Rohan Vikas

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

Class SYMCA Batch B1 Performed on

Roll No. C Expt. No. Submitted on

Remarks Returned on

Procedure U(i,j)

Description :- Replace the disjoint sets with root i,j, i≠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 & j i≠j, using the weighting rule.

PARENT(i) = −COUNT(i) &

PARENT (i) = − COUNT (j)

Declaration :- integer i,j,x

x ← PARENT(i) + PARENT (j)

if (PARENT (i) > PARENT (j)) then

PARENT (i) ← j;

PARENT (j) ← x;

else

PARENT(j) ← i;

PARENT (i) ← x;

Listing

```
    endif
end UNION
procedure FIND (i)
  Description :- Find the root of the tree containing
                 i, use the collapsing rule to collapse all n
                 i to the root
  Declaration :- integer i, j, k
    j ← i  //find first root of tree
  while PARENT (j) > 0
        j ← PARENT (j);
   repeat
   return(j) 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 Badhe Rohan Vikas
Expt. Title Write a program to implement union & find operation.
Class SYMCA Batch B1 Performed on
Roll No. 04 Expt. No. Submitted on
Remarks Returned on

Modify union:-

Procedure UNION (i,j)

Description:-

UNION set with roots i & j, i ≠ j
using the weighting Rule
PARENT (i) ← -COUNT (i) &
PARENT (j) ← -COUNT (j)

Declaration:-

Integer i, j, x

Algorithm:-

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.

// modify FIND :-

Procedure FIND(i)

Description :-

Find the root of tree containing element i using the collapsing rule for collapsed.

Declaration :-

integer i, j, k

Algorithm :-

// Find first Tist a root of tree

while PARENT (j) > 0, do

j ← PARENT (j)

repeat

return (j)

// collapse the nodes from node (i) up to the root

k ← i

while k ≠ j, do

temp ← PARENT (k)

PARENT (k) ← j

k ← Temp

repeat

return (j)

END FIND.

DEPARTMENT OF COMPUTER SCIENCE
INSTITUTE OF MANAGEMENT AND RESEARCH, JALGAON

Name Badhe Rohan Vikas

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

Class SYMCA Batch B1 Performed on

Roll No. 04 Expt. No. Submitted on

Remarks Good Returned on

write an Algorith to find minimum & maximum from given array.

Procedure MAXMIN (P,q,max,min)

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

Declaration :- integer P, q

$1 \leq P \leq q \leq n$

```
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
|
|
```

```
        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 Badhe Rohan Vikas

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

Class SYMCA Batch B1 Performed on

Roll No. 05 Expt. No. Submitted on

Remarks Good Returned on

BINARY SEARCH :-

Procedure BINSEARCH (A,n,x,j)

Description :- Given an Array A[1:n] of elements in nondecrasing order n>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
    end case
repeat
j ← 0
```

END BINSEARCH.

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_SEARCH 1
```

DEPARTMENT OF COMPUTER SCIENCE
INSTITUTE OF MANAGEMENT AND RESEARCH, JALGAON

Name Badhe Rohan Vikas
Expt. Title Write a program for creating Heap Sort (Ascending)
Class SYMCA Batch B1 Performed on
Roll No. 04 Expt. No. Submitted on
Remarks Ok Returned on

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

Procedure HEAP_SORT(A,n)

Description :-

This procedure sort the n elements of A(1:n). Heap sort rearrange them in-place into non-decreasing order. where A(1:n) is array contain '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 & 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

```
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.

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 Badhe Rohan Vikas
Expt. Title Write a program for creating Heap-sort (Descending)
Class SYMCA Batch B1 Performed on
Roll No. 04 Expt. No. Submitted on
Remarks Good 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 them 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 & 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

```
   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 Badhe Rohan Vikas
Expt. Title Write a Program for meerge Sort.
Class SYMCA Batch B1 Performed on
Roll No. 04 Expt. No. Submitted on
Remarks Returned on

Write an algorithm for meerge sort

Procedure MERGE_SORT (low, high)

Description :- A [low : high] is 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 ~~subject~~ 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
while i ≤ mid & 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 Badhe Rohan Vikas
Expt. Title Write a program for quick sort (Ascending)
Class SYMCA Batch B1 Performed on
Roll No. 04 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 & must be ≥ all the elements in a(1:n)

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

```
If (p<a) then
    j← PARTITON (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 & a(k) ≥ t for q < k < P. q is returned set a(P) = ∞

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
  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 Badhe Rohan Vikas
Expt. Title
Class MCA III Batch B1 Performed on
Roll No. 04 Expt. No. Submitted on
Remarks Returned on

write Algorithm to find solution of KNAPSACK instant

```
Procedure GREEDY_KNAPSACK (P, W, M, X, n)
  Description: P(1:n) & W(1:n) contain
  the Profit & weights respectively of n objects
  ordered so that P(i)/W(i) >= P(i+1)/W(i+1) M
  is the KNAPSACK size and (1:n) is the solution
  vector.
  // Assuming that data given is sorted according to
    P/W as describe above
  Declaration:-
    P(1:n), W(1:n), x(1:n), M, C integer i, n
    x ← 0
    Cu ← M
    for i ← 1 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
```

DEPARTMENT OF COMPUTER SCIENCE
INSTITUTE OF MANAGEMENT AND RESEARCH, JALGAON

Name Badhe Rohan Vikas
Expt. Title Write a Program to find Shortest Path using single source shortest path
Class SYMCA Batch B1 Performed on
Roll No. 04 Expt. No. Submitted on
Remarks Returned on

Write a algorithm to find ~~So~~shortest path using single source shortest path

Procedure ~~So~~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 (4) = min {DIST (w) & S(w)=0}
S(u) ← 1 // put u in S.
```

```
for (each w adjacent to v with S(w) = 0) do
    // update distance
    if (DIST(w) > DIST(u) + COST(u,w)) then
        DIST(w) ← DIST(u) + COST(u,w);
    end if
  repeat
repeat
end SHORTEST-PATH
```

DEPARTMENT OF COMPUTER SCIENCE
INSTITUTE OF MANAGEMENT AND RESEARCH, JALGAON

Name Badhe Rohan Vikas
Expt. Title write a program to find minimum-cost Spanning Trees
Class SYMCA Batch B2 Performed on
Roll No. 04 Expt. No. Submitted on
Remarks Good Returned on

write a Algorithm to find minimum-cost Spanning Trees (Prim's & Krukal'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 + ∞ if no edge (i,j) exists. A ~~minimum~~ minimum spanning tree is computed & stored as a set of edges in the array T(1:n,2), T(i,2), 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, 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
```

```
NEAR(k) ← 0
NEAR(j) ← 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 Badhe Rohan Vikas
Expt. Title write a program to find Shortest path using all Pair Path
Class SY BMCA Batch B1 Performed on
Roll No. 04 Expt. No. Submitted on
Remarks Ge 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 vi to vj 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
```

INSTITUTE OF MANAGEMENT AND RESEARCH, JALGAON

Name Badhe Rohan Vikas

Expt. Title Write a program to find logest Common Sub-sequence

Class SYMCA Batch B1 Performed on

Roll No. 04 Expt. No. 9 Submitted on

Remarks Returned on

Procedure LCE-LENGTH (x, Y)

Description :- $x = (x_1, x_2 \dots x_m)$ &
$Y = (y_1, y_2 \dots y_n)$ are the two given sequence.
The algo uses two m x n matrices, C(0:m, 0:n) & B(0:m, 0:n) matrix stores the length of ~~the~~ ICS & matrix B stores symbols S which are we 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
    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(a: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 ← i-1 & j ← j+1
        endif
      endif
    repeat
  repeat
  for i ← top to 1 do
      PRINT (STACK(i))
  repeat
END LCS_PRINT
```

DEPARTMENT OF COMPUTER SCIENCE
INSTITUTE OF MANAGEMENT AND RESEARCH, JALGAON

Name Badhe Rohan Vikas

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

Class SYMCA Batch B1 Performed on

Roll No. 04 Expt. No. Submitted on

Remarks Returned on

Breath 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 ADD Q(w,Q)
         VISITED(w) ← 1
      endif
   repeat
   if Q is empty then
      return
   end if
   Call DELETE Q(u,Q)
repeat
end BFS
```

DEPARTMENT OF COMPUTER SCIENCE
INSTITUTE OF MANAGEMENT AND RESEARCH, JALGAON

Name Badke Rohan Vikas

Expt. Title write a Program to implement breath & depth first search

Class SYMCA Batch B1 Performed on

Roll No. 04 Expt. No. Submitted on

Remarks Ok Returned on

Breath 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 VISITE 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 VISITE D(w) = 0, then
            call ADD Q(w,Q)
            VISITED(w) ← 1
        end if
    repeat
    if Q is empty then
        return
    end if
    Call DELETE Q(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 Badhe Rohan Vikas

Expt. Title write a program to find all solutions for 8-quen problem using back tracking

Class SYMCA Batch B1 Performed on

Roll No. Expt. No. Submitted on

Remarks Ve Returned on

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

```
Procedure NQUEENS(n)
    Description :- using backtracking, this procedure
      prints all posible placement 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 & pas
    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
```

Procedure PLACE(K)

Description :- return ~~true~~ true if a queen can be pl
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