Program

Q Write a program to find maximum Depth or Height of a tree

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
struct node
{
    int data;
    struct node * left;
    struct node* right;
};
int maxDepth (struct node * node)
{
    if (node == NULL)
        return 0;
    else
    {
    int lDepth = maxDepth (node -> left);
    int rDepth = maxDepth (node -> right);
    if (lDepth > rDepth)
        return (lDepth + 1);
    else
        return (rDepth + 1);
    }
struct node* newnode (int data)
    {
        struct node *node = (struct node*) malloc
                (sizeof (struct node));
```

OUTPUT

Height of tree is: 4

```
    node -> data = data;
    node -> left = NULL;
    node -> right = NULL;
    return (node);
  }
int main ()
{
 clrscr();
struct node *root = newnode (5);
 root -> left = newnode (2);
 root -> right = newnode (8);
 root -> left -> left = newnode (14);
 root -> left -> left -> left = newnode (24);
 root -> left -> right = newnode (25);
 printf(" Height of tree is %d", maxDepth(root));
  getchar();
  return 0;
  }
```

Program

Write a program for traversing a graph through DFS.

```
#include <stdio.h>
#include <conio.h>
#define MAX 20

typedef enum boolean {false, true} bool;
int adj[MAX][MAX];
bool visited[MAX];
int n;
void main()
{   int i, v, choice;
    clrscr();
 create_graph();
  while(1)
  {
   printf("1. Adjancency matrix \n");
   printf("2. DFS using stack \n");
   printf("3. DFS through recursion \n");
   printf("4. No. of components \n");
   printf("5. Exit");
   printf("Enter your choice:");
   scanf("%d", &choice);
  switch(choice)
  {
case 1:
printf("Adjacency Matrix \n");
    display();
```

OUTPUT

```
Enter no. of nodes 2
Enter edge 1(0 0 to quit) : 1
2
Enter edge 2(0 0 to quiet): 1
2

1. Adjacency Matrix
2. DFS using stack
3. DFS through recursion
4. No. of components
5. Exit
Enter your choice 2
Enter the starting node 5
5
```

```
    break;
Case 2:
printf("Enter starting node : ");
    scanf("%d", &v);
    for(i=1; i<=n; i++)
        visited[i] = False;
    dfs(v);
    break;
Case 3:
printf("Enter starting node:");
scanf("%d", &v);
for(i=1; i<=n; i++)
    visited[i] = false;
    dfs_rec(v);
    break;
Case 4:
printf("Components are");
    components();
    break;
Case 5:
exit(1);
default:
printf("Enter correct choice\n");
    break;
}}}

create_graph()
{   int i, max_edge, origin, dest;
    printf("Enter no. of nodes:");
```

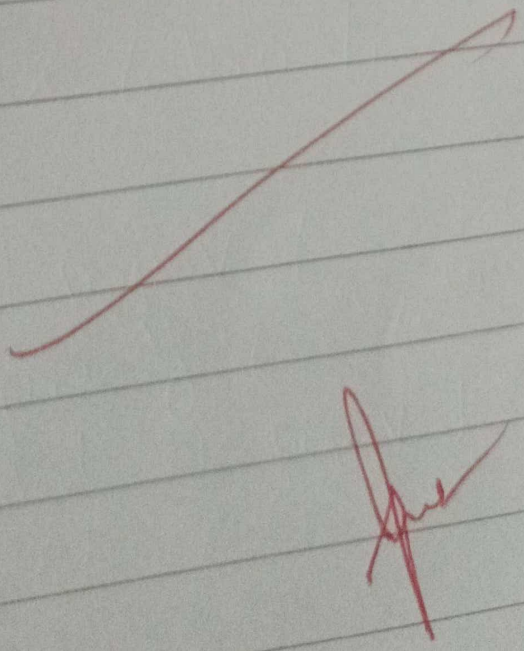
```
array ("%d", &n)
max_edges = n*(n-1);
for(i=1; i<=maxedges; i++)
{
    printf("Enter edge %d (00 to quit) :",i);
    scanf("%d %d", &origin, &dest);
    if((origin==0) && (dest==0))
    break;
if(origin >n ||dest >n || origin <=0||dest <=0)
{   printf("Invalid edge!\n");
    i--;
}
else
{   adj[origin][dest] = 1;
}}
return 0;
} display()
} int i,j;
  for(i=1; i<=n; i++)
{ for(j=1; j<=n; j++)
  printf("%d", adj[i][j]);
  printf("\n");
} return 0;
}

dfs_rec(int v)
{ int i;
  visited[v] = true;
  printf("%d", v);
```

```
for (i = 1; i <= n; i++)
    if (adj[v][i] == 1 && visited[i] == false)
        dfs_rec(i);
return 0;
} dfs (int v)
{   int i, stack[MAX], top = -1, pop_v, visited[MAX];
    int ch;
    top++;
    stack[top] = v;
    while (top >= 0)
    {
    pop_v = stack[top];
     top--;
     if (visited[pop_v] == false)
     {        printf("%d", pop_v);
              visited[pop_v] = true;
     }
else
     cout continue;
for (i = n; i >= 1; i--)
{   if (adj[pop_v][i] == 1 && visited[i] == false)
    {  top++;
       stack[top] = i;
}}}
return 0;
}
```

Topic Date

```
components ()
{   int i;
  for(i=1; i<=n; i++)
     visited [i] = false;
  for(i = 1; i <= n; i++)
  {
     if (visited [i] == false)
        dfs_rec(i);
  } printf ("\n");
return 0;
   }
```



Program

Write a program for traversing a graph through BFS.

```
#include <stdio.h>
#include <conio.h>
#def MAX 20
typedef enum boolean {false, true} bool;
int adj [MAX][MAX];
bool visited [MAX];
int n;
void main()
{
  int i, v, choice;
  clrscr();
create_graph();
  while(1)
  { printf("1. Adjacency Matrix");
    printf("2. BFS \n");
    printf("3. Adjacent vertices\n");
    printf("4. Exist \n");
  printf("Enter your choice:");
  scanf("%d", &choice:");
    switch(choice)
    {
case 1:
printf("Adjacency Matrix \n");
  display();
    break;
case 2:
printf("Enter starting node for BFS:");
```

OUTPUT

1. Adjacency matrix
2. BFS
3. Adjacent vertices
4. Exit

Enter your choice

```
scanf("%d", &v);
for (i = 1; i <= n; i++)
    visited[i] = false;
bfs(v);
break;
Case 3:
printf("Enter node to find Adjacent vertices: ");
scanf("%d", &v);
printf("Adjacent vertices are:");
adj_nodes(v);
break;
Case 4:
exit(1);
default:
printf("Wrong choice \n");
break;
}}}
create_graph()
{
int i, max_edge, origin, dest;
printf("Enter no. of ~~choice~~ nodes: ");
scanf("%d", &n);
max_edges = n*(n-1);
for (i=1; i<=max_edges; i++)
{
printf("Enter edges %d(00 to quit): ", i);
scanf("%d%d", &origin, &dest);
if((origin == 0) && (dest == 0))
break;
if(origin > n || dest > n || origin <= 0 || dest <= 0)
```

```
{ printf(" Invalid edge ! \n");
    i-- ;
}
else
 {  adj[origin][dest]=1;
 }}
 return 0;
  }
display()
 {  int i, j;
    for(i=1; i<=n; i++)
   {     for (j=1; j<=n; j++)
     printf("%4d", adj[i][j]);
          printf("\n");
}  return 0;
        }
bfs(int v)
{
   int i, front, rear;
    int que[20];
    front = rear=-1;
printf("%d", v);
 visited[v] = true
  rear++;
  front++;
   que[rear]=v;
while (front <= rear)
 {
```

```
    v = que[front];
    front++;
    for(i=1; i<=n; i++)
    {
      if(adj[v][i]==1 && visited[i]==0)
      {
        printf("%d", i);
        visited[i] = true;
        rear++;
        que[rear] = i;
      }
    }
    return 0;
}

adj_nodes(int v)
{
    int i;
    for(i=1; i<=n; i++)
if(adj[v][i]==1)
    printf("%d", i);
printf("\n");
    return 0;
}
```

Topic Date

35

Ques W.A.P to implement the merge sort.

```
# include <stdio.h>
# include <conio.h>
  int array [20];
void merge (int low, int mid, int high)
{
   int temp [20];
   int j = mid + 1;
   int i = low;
   int k = low;
while ((i<= mid) && (j <= high))
  {   if (array [i] <= array [j])
      temp [k++] = array [i++];
      else
       temp [k++] = array [j++];
  }
  while (i <= mid)
   temp [k++] = array [i++];
    while ( j <= high)
  temp [k++] = array [j++];
   for ( i= low; i <=high ; i++)
    array [i] = temp[i];
  }
   void merge_sort (int low, int high)
   {     int mid;
        if ( low != high)
  {    mid = (low + high)/2;
       merge_sort (low, mid);
```

OUTPUT

Enter the no of element: 4
Enter element 1: 1
Enter element 2: 6
Enter element 3: 4
Enter element 4: 9

The sorted list is
1 6 4 9

Sorted list is:-
1 4 6 9

```
    merge_sort(mid+1, high);
    merge(low, mid, high);
  }}
  void main()
{   int i, n;
    clrscr();
    printf("\n Enter the no. of elements:");
    scanf("%d", &n);
{   for(i=0; i<n; i++)

    printf("\n Enter element %d:", i+1);
    scanf("%d", &array[i]);
}
  printf("\n Unsorted list is:\n");
    for(i=0; i<n; i++)
    printf("%d\t", array[i]);
    merge_sort(0, n-1);
  printf("\n Sorted list is:\n");
      for(i=0; i<n; i++)
printf("%d\t", array[i]);
    getch();
  }
```

Ques W.A.P to implement the quicksort.

```
#include <stdio.h>
#include <conio.h>
void quicksort (int a[], int low, int high);
int partition (int a[], int low, int high);
int a[5] = {55, 1, 78, 13, 45};
void main()
{
    clrscr();
    int i, n;
    printf("\n Original array ");
    for(i=0; i<5; i++)
        printf("%4d", a[i]);
    quicksort(a, 0, 4);
    printf("\n The sorted array is \n");
    for(i=0; i<5; i++)
        printf("%4d", a[i]);
    getch();
}
void quicksort (int a[], int low, int high)
{
    int j;
    if(low < high)
    {
        j = partition(a, low, high)
        quicksort(a, low, j-1);
        quicksort(a, j+1, high);
    }
}
```

OUTPUT

Original array = 55 1 78 13 9

The sorted array is

1 13 45 55 78

```
int partition (int a[], int low, int high)
{
    int i, j, temp, key;
    key = a[low];
    i = low + 1;
    j = high;
    while (1)
    {
while (i<high && key >= a[i])
    i++;
    while (key<a[j])
    j--;
    if (i<j)
{
    temp = a[i];
    a[i] = a[j];
    a[j] = temp;
}
    else
    {   temp = a[low];
    a[low] = a[j];
    a[j] = temp;
    }
    return j;
}}
```

Topic _____ Date _____

Page 37

WAP to implement Heap Sort using array

```
#include <stdio.h>
#include <conio.h>
int a[20], n;
void display()
{
    int i;
    for (i=0; i<n; i++)
    printf ("%d", a[i]);
    printf ("\n");
}
void insert (int num, int loc)
{
    int parent;
    while (loc>0)
    {
    parent = (loc-1)/2
    if (num <= a[parent])
    {
        a[loc] = num;
        return;
    }
    a[loc] = a[parent];
    loc = parent;
    }
    a[0] = num;
}
void create_heap()
{
    int i;
    for (i=0; i<n; i++)
    insert (a[i], i);
```

OUTPUT

```
Enter no. of elements : 3
Enter element 1 : 4
Enter element 2 : 6
Enter element 3 : 10

Entered list is :-
4 6 10

Heap list
10 6 4

Sorted list is
4 6 10
```

```
}
void del_root(int last)
{
   int left, right, i, temp;
      i = 0;
   temp = a[i];
   a[i] = a[last];
   a[last] = temp;
   left = 2*i+1; /* left child */
   right = 2*i+2; /* right child */
while (right < last)
{
   if (a[i] >= a[left] && a[i] >= a[right])
   return;
   if (a[right] <= a[left])
   {
      temp = a[i];
      a[i] = a[left];
     a[left] = temp;
      i = left;
   }
   else
   {     temp = a[i];
         a[i] = a[right];
         a[right] = temp;
          i = right;
   }
   left = 2*i+1;
```

```
    right = 2*i+2;
  }
  if(left == last-1 && a[i] < a[left])
  {
      temp = a[i];
      a[i] = a[left];
      a[left] = temp;
  }}
void heap_sort()
{
    int last;
    for(last=n-1; last>0; last--)
    del_root(last);
}
Void main()
{
    int i;
  printf("\n Enter no. of elements:");
   scanf("%d", &n);
    for(i=0; i<n; i++)
{
     printf("\n Enter Element %d:", i+1);
    scanf("%d", &a[i]);
}  printf("\n Entered list is :\n");
    display();
  create_heap();
  printf("\n Heap is :\n");
   display();
   heap_sort();
     printf("\n Sorted list is: \n");
     display();
    getch();
}
```

OUTPUT

How many elements you want to enter in the array :- 3
Enter element 1 : 4
Enter element 2 : 6
Enter Element 3 : 10
Enter the element to be searched : 10
10 found at Position 3

Ques W.A.P to implement the linear search

```
#include <stdio.h>
#include <conio.h>
   void main()
{
  char ch;
  int arr[50], n, i, item;
  clrscr();
printf("\n How many element enter in the array :");
scanf("%d", &n);
{for (i=0; i<n; i++)
        printf("\n Enter element %d:", i+1);
         scanf("%d", &arr[i]);
  }

printf("\n Enter the element to be searched:");
scanf("%d", &item);
for (i=0; i<n; i++)
        if (item == arr[i])
{   printf("\n%d found at position %d", item, i+1);
   break;
}}
  if (i==n)
printf("\n Item %d not found in array \n", item);
getch();
   }
```

Topic

Date

Ques W.A.P to implement the binary search. 13

```
#include <stdio.h>
#include <conio.h>
void main()
{
    char ch;
    int arr[20], start, end, mid, n, i, data;
clrscr();
printf("\n How many elements you want to enter:");
scanf("%d", &n);
for(i=0; i<n; i++)
{
    printf("\n Enter element %d", i+1);
    scanf("%d", &arr[i]);
}
printf("Enter the element to be searched:");
scanf("%d", &data);
start=0;
end=n-1;
mid=(start+end)/2;
while(data!=arr[mid] && start<=end)
{
    if(data>arr[mid])
        start=mid+1;
    else
    end=mid-1;
    mid=(start+end)/2;
}
if(data==arr[mid])
```

OUTPUT

```
How many elements you want to Enter: 3
Enter Element 1: 3
Enter Element 2: 5
Enter Element 3: 4
Enter the element to be searched: 5
5 Found at Position 2
```

```
printf ("\n\t%d found at position %d \n", data, mid+1);
    if (start > end)
        printf ("\n\t%d not found in array \n", data);
    getch();
}
```