ADA-LAB-PROGRAMS

1. Design a quicksort program in 'C' to arrange a given array of 'n' real numbers so that all its negative elements precede all its positive elements. Find its time complexity.

```
#include<stdio.h>
#include<time.h>
int a[50],n;
int main()
{
        int i;
        clock_t f,s;
        printf("\n\t\t\t QUICK SORT \n");
        printf("\n Enter the length of the array:");
        scanf("%d",&n);
        printf("\n Enter elements of the array \n");
        for(i=1;i<=n;i++)
                scanf("%d",&a[i]);
        s=clock();
        quick_sort(1,n);
        f=clock();
        printf("\n Sorted elements are \n");
        for(i=1;i<=n;i++)
                printf("%d\t",a[i]);
        printf("Time taken to execute quick sort is %f",(f-s));
        return 0;
 }
quick_sort(int p,int q)
{
        int j;
        if(p < q)
        {
                j=q+1;
                j=partition(p,j);
                quick_sort(p,j-1);
                quick_sort(j+1,q);
        }
}
partition(int m,int p)
{
        int i,t,v;
        v=a[m];
        i=m;
        while(i<p)
        {
                do i++;
                while(a[i]<v);
                do p--;
                while(a[p]>v);
                if(i<p)
                {
                        t=a[i];
                        a[i]=a[p];
```

```
                        a[p]=t;
                }
                else
                        break;
        }
        a[m]=a[p];
        a[p]=v;
        return p;
}
```

2. Consider a list of 'n' files numbered using ID's. Write a C program to sort the files based on its ID using merge sort. Also find its time complexity.

```
#include<stdio.h>
int n,a[50];

main()
{
        int i;
   printf("\n\t\t\t MERGE SORT \n");
        printf("\n Enter the size of the array:");
        scanf("%d",&n);
        printf("\n Enter the array elements \n");
        for(i=1;i<=n;i++)
                scanf("%d",&a[i]);
   divide(1,n);
   printf("\n Sorted elements of the array \n");
        for(i=1;i<=n;i++)
                printf("%d\t",a[i]);
}

divide(int low,int high)
{
        int mid;
        if(low < high)
        {
                mid=(low+high)/2;
                divide(low,mid);
                divide(mid+1,high);
                mergesort(low,mid,high);
        }
}

mergesort(int low,int mid,int high)
{
        int i,h,j,k,b[50];
        i=h=low;
        j=mid+1;
        while(h<=mid && j<= high)
        {
                if(a[h]<=a[j])
                {
                        b[i]=a[h];
                        h=h+1;
                }
```

```
                else
                {
                        b[i]=a[j];
                        j=j+1;
                }
                i=i+1;
        }
        if(h>mid)
                for(k=j;k<=high;k++)
                {
                        b[i]=a[k];
                        i=i+1;
                }
        else
                for(k=h;k<=mid;k++)
                {
                        b[i]=a[k];
                        i=i+1;
                }
        for(k=low;k<=high;k++)
                a[k]=b[k];
}
```

3. Write a C program that, for a given digraph outputs all the vertices reachable from a given starting vertex using BFS method.

```
#include<stdio.h>
int a[20][20],q[20],visited[20],n,i,j,f=0,r=-1;
void bfs(int v)
{
        for (i=1;i<=n;i++)
                if(a[v][i] && !visited[i])
                        q[++r]=i;
        if(f<=r)
        {
                visited[q[f]]=1;
                bfs(q[f++]);
        }
}
void main()
{
        int v;
        printf("\n Enter the number of vertices:");
        scanf("%d",&n);
        for (i=1;i<=n;i++)
        {
                q[i]=0;
                visited[i]=0;
        }
        printf("\n Enter graph data in matrix form:\n");
        for (i=1;i<=n;i++)
                for (j=1;j<=n;j++)
                        scanf("%d",&a[i][j]);
        printf("\n Enter the starting vertex:");
        scanf("%d",&v);
   bfs(v);
        printf("\n The node which are reachable are:\n");
```

```
        for (i=1;i<=n;i++)
                if(visited[i])
                        printf("%d\t",i);
      else
                        printf("\n Bfs is not possible");
}
```

4. Consider a network having 'n' systems. Design a DFS based program in 'C' which outputs all systems reachable from a given system

```
#include<stdio.h>
int a[20][20],reach[20],n;
void dfs(int v)
{
        int i;
        reach[v]=1;
        for (i=1;i<=n;i++)
                if(a[v][i] && !reach[i])
                {
                        printf("\n %d->%d",v,i);
                        dfs(i);
                }
}
void main()
{
        int i,j,v,count=0;
        printf("\n Enter number of vertices:");
        scanf("%d",&n);
        for (i=1;i<=n;i++)
        {
                reach[i]=0;
                for (j=1;j<=n;j++)
                        a[i][j]=0;
        }
        printf("\n Enter the adjacency matrix:\n");
        for (i=1;i<=n;i++)
                for (j=1;j<=n;j++)
                        scanf("%d",&a[i][j]);
        printf("Enter the source vertex:\n");
   scanf("%d",&v);
   dfs(v);
        printf("\n");
        for (i=1;i<=n;i++)
                if(reach[i])
                        count++;
        if(count==n)
                printf("\n Graph is connected");
        else
                printf("\n Graph is not connected");
}
```

5. Suppose you are given a list of students who are assigned IDs. Write a C program to sort these students based on their id's using heapsort.

```
#include <stdio.h>

void main()
{
	int a[100], n, i;
	printf("Enter the number of elements \n");
	scanf("%d",&n);

	printf("Enter the elements \n");
	for(i=0;i<n;i++)
		scanf("%d",&a[i]);

	heapSort(a,n);

	printf("Sorted elements are \n");
	for(i=0;i<n;i++)
		printf("%d ",a[i]);
}

heapSort(int a[], int n)
{
	int i, temp;

	for (i = n-1; i >= 0; i--)
		siftDown(a, i, n - 1);

	for (i=n-1;i>=1;i--)
	{
		temp = a[0];
	a[0] = a[i];
	a[i] = temp;
	siftDown(a, 0, i-1);
	}
}


siftDown(int a[], int root, int bottom)
{
	int done, maxChild, temp;

	done = 0;
	while ((root*2 <= bottom) && (!done))
	{
	if (root*2 == bottom)
		maxChild = root * 2;
	else if (a[root * 2] > a[root * 2 + 1]) //left child greater than right child
		 maxChild = root * 2;
	else
		maxChild = root * 2 + 1;

	if (a[root] < a[maxChild]) //exchange root with maxchild
	{
		temp = a[root];
		a[root] = a[maxChild];
		a[maxChild] = temp;
		root = maxChild;
	}
	else
	done = 1;
	}
```

```
}
```

6. Consider the problem of searching for genes in DNA sequences. A DNA sequence is represented by a text using alphabets [A, C, G, T]. Design a 'C' program to locate a pattern in a given DNA sequence using Horsepool's algorithm.

```
#include <string.h>
#include <stdio.h>

char str[100],ptn[20];
int res,m,n,len,len1,i,j,k,table[1000];

void main()
{
        printf("Enter the text \n");
        gets(str);
        printf("Enter the pattern to be found \n");
        gets(ptn);
        res=horspool(ptn,str);
        if(res==-1)
        printf("\nPattern not found\n");
        else
        printf("Pattern found at %d position \n",res+1);
}

void shift(char p[])
{
        len=strlen(p);
        for(i=0;i<1000;i++)
                table[i]=len;
        for(j=0;j<=len-2;j++)
                table[p[j]]=len-1-j;
}

int horspool(char p[], char t[])
{
        shift(p);
        m=strlen(p);
        n=strlen(t);
        i=m-1;
        while(i<=n-1)
        {
                k=0;
                while(k<=m-1 && (p[m-1-k]==t[i-k]))
                k++;
                if(k==m)
        return i-m+1;
                else
        i=i+table[t[i]];
        }
        return -1;
}
```

7. Consider a network of 'n' systems represented as a Graph. Write a 'C' program to find the transitive closure of such a network using warshall's algorithm

```
#include <stdio.h>
void warshall (int a[][20], int n)
{
	int k,i,j;
	for(k=0; k<n; k++)
		for(i=0; i<n; i++)
			for(j=0; j<n; j++)
				a[i][j] =( a[i][j] || a[i][k] && a[k][j]);
}

void main()
{
	int n,a[20][20],i, j ;
	printf("\n enter number of nodes in the graph \n ");
	scanf("%d", &n);
	printf("Enter the adjacency matrix for the graph:\n");
	for(i=0; i<n; i++)
		for(j=0; j<n; j++)
			scanf("%d", &a[i][j]);
	warshall(a,n);
	printf(" the path matrix is \n");
	for (i=0; i<n; i++)
	{
		for (j = 0; j<n; j++)
			printf("%d\t" , a[i][j]);
		printf(" \n");
	}
}
```

8. Suppose in a network of cities, you are interested in finding shortest paths between all cities. Design a 'C' program to implement this using Floyd's algorithm.

```
#include <stdio.h>
void floyd (int a[20][20], int n)
{
	int k,i,j;
	for(k=0;k<n;k++)
		for(i=0;i<n;i++)
			for(j=0;j<n;j++)
				if(a[i][j] > (a[i][k] + a[k][j]))
				a[i][j] = (a[i][k] + a[k][j]);
	printf(" All Pairs Shortest Paths:\n");
	for(i=0;i<n;i++)
	{
		for (j=0;j<n;j++)
			printf("%d\t" , a[i][j]);
		printf("\n");
	}
}

void main()
{
	int n,a[20][20],i, j ;
	printf("Enter the number of vertices\n");
	scanf("%d", &n);
```

```
	printf("Enter the cost adjacency matrix for the graph [999 for no edge, 0 for self loops]: \n");
	for (i=0;i<n;i++)
		for (j=0;j<n;j++)
			scanf("%d", &a[i][j]);
	floyd(a,n);
}
```

9. Suppose a travel agent is interested in finding shortest path from a single city to all the other cities in a network of 'n' cities. Write a C program to implement this using Dijkstra's algorithm.

```
#include <stdio.h>
void main()
{
	int n,i,j,a[10][10],s[10],d[10],v,k,min,u;
	printf("Enter the number of vertices\n");
	scanf("%d",&n);
	printf("Enter the cost matrix \n");
	printf("Enter 999 if no edge between vertices \n");
	for(i=1;i<=n;i++)
		for(j=1;j<=n;j++)
			scanf("%d",&a[i][j]);
	printf("Enter the source vertex \n");
	scanf("%d",&v);
	for(i=1;i<=n;i++)
	{
		s[i]=0;
		d[i]=a[v][i];
	}
	d[v]=0;
	s[v]=1;
	for(k=2;k<=n;k++)
	{
		min=999;
		for(i=1;i<=n;i++)
		if(s[i]==0 && d[i]<min)
		{
			min=d[i];
			u=i;
		}
		s[u]=1;
		for(i=1;i<=n;i++)
			if(s[i]==0)
			{
				if(d[i]>(d[u]+a[u][i]))
				d[i]=d[u]+a[u][i];
			}
	}
	printf("The shortest distance from %d is \n",v);
	for(i=1;i<=n;i++)
		printf("%d-->%d=%d\n",v,i,d[i]);
}
```

10. Consider a Electrical layout where 'n' houses are connected by electrical wires. Design a 'C' program using Prim's algorithm to output a connection with minimum cost.

```
#include <stdio.h>
int n,c[20][20],i,j,visited[20];

void main()
{
	printf("Enter number of vertices \n");
	scanf("%d",&n);
	printf("Enter the cost matrix \n");
	printf("Enter 999 if no direct edges \n");
	for(i=1;i<=n;i++)
	{
		for(j=1;j<=n;j++)
		scanf("%d",&c[i][j]);
		visited[i]=0;
	}
	prim();
}

prim()
{
	int min,b,a,k,count=0,cost=0;
	min=999;
	visited[1]=1; /* 1st vertex is visited */
	while(count<n-1)
	{
		min=999;
		for(i=1;i<=n;i++)
			for(j=1;j<=n;j++)
				if(visited[i] && !visited[j] && min>c[i][j]) /* if i is visited but j is not
visited and c[i][j] < min*/
				{
					min=c[i][j]; /* assign c[i][j] as minimum cost*/
					a=i;
					b=j;
				}
		printf("%d--->%d = %d\n",a,b,c[a][b]); /* prints each edge in the MST and its cost */
		cost+=c[a][b]; /* adds the minimum cost */
		visited[b]=1;
		count++;
	} //end while
	printf("Total cost of Spanning tree is %d",cost); /* cost holds the minimum cost of the MST */
}
```

11. A Government wants to construct a road network connecting 'n' towns. Suppose each road must connect '2' towns and be straight. Write a C program using Kruskal's algorithm to output the least expensive tree of roads.

```
#include <stdio.h>
int min, cost[100][100],parent[100],i,j,x,y,n;

void main()
{
	int count=0,tot=0,flag=0;
	printf("Enter the number of vertices \n");
	scanf("%d",&n);
	printf("Enter the cost matrix \n");
	printf("Enter 999 for no edges and self loops \n");
	for(i=1;i<=n;i++)
```

```
                for(j=1;j<=n;j++)
                {
                        scanf("%d",&cost[i][j]);
                        parent[j]=0;
                }
        while(count!=n-1 && min!=999)
        {
                find_min();
                flag=check_cycle(x,y);
                if(flag==1)
                {
                        printf("\n%d----->%d==%d\n",x,y,cost[x][y]);
                        count++;
                tot+=cost[x][y];
                }
                cost[x][y]=cost[y][x]=999;
        }
        printf("\nThe total cost of minimum spanning tree=%d",tot);
}

find_min()
{
        min=999;
        for(i=1;i<=n;i++)
                for(j=1;j<=n;j++)
                        if(cost[i][j]<min)
                        {
                                min=cost[i][j];
                                x=i;
                                y=j;
                        }
}

int check_cycle(int x,int y)
{
        if((parent[x]==parent[y]) && (parent[x]!=0))
                return 0;
   else if (parent[x]==0 && parent[y]==0)
                parent[x]=parent[y]=x;
   else if(parent[x]==0)
                parent[x]=parent[y];
   else if(parent[x]!=parent[y])
                parent[y]=parent[x];
   return 1;
}
```

12. Consider ‘n’ patients and ‘nxn’ small rooms. Design a C program to allot the patients to these rooms using nqueen’s method such that no two patients are allotted rooms in same row, column or diagonal.

```
#include <stdio.h>
#include <math.h>
int count=0,x[100];
main()
{
```

```
	int n;
	printf("\t\t\tN QUEEN'S PROBLEM\n");
	printf("\nEnter the number of queens:");
	scanf("%d",&n);
	nqueen(1,n);
	if(count==0)
		printf("\n There is no solution for '%d - Queens' problem",n);
	else
		printf("Total number of solutions :%d",count);
}

int  place(int k,int i) // checks whether kth queen can be placed in ith column
{
	int j;
	for(j=1;j<k;j++)   // x[j]=i means same column, (x[j]-j) =( j-k) means same diagonal
	if((x[j]==i) ||(abs(x[j]-i)==abs(j-k)))
		return(0); // kth queen cannot be placed in ith column
	return(1);
}

nqueen(int k,int n)
{
	int i,j,p;
	for(i=1;i<=n;i++)
	if(place(k,i)) // if kth queen can be placed in ith column
	{
		x[k]=i; // place kth Queen in ith column
		if(k==n) // if all queens are placed then we print the solution matrix
		{
			count++;
			for(j=1;j<=n;j++)
			{
				for(p=1;p<=n;p++)
					if(x[j]==p)
						printf(" q ");
					else
						printf(" 0 ");
				printf("\n");
			}
		}
		else
			nqueen(k+1,n);
	}
	printf("\n");
}
```