*ASSIGNMENT-12*

1. ***WAP to implement the following:***

***a. Bubble sort.***

#include<stdio.h>

#include<stdlib.h>

#define SIZE 100

void bubble\_sort(int \*arr,int n);

int main()

{

int arr[SIZE];

printf("Enter size of array : ");

int n;

scanf("%d",&n);

printf("Enter elements of array : ");

int i;

for(i=0;i<n;i++)

scanf("%d",&arr[i]);

bubble\_sort(arr,n);

return 0;

}

void bubble\_sort(int \*arr,int n)

{

int i,j,temp;

for(i=0;i<n;i++)

for(j=0;j<n-i-1;j++)

if(arr[j]>arr[j+1])

{

temp = arr[j];

arr[j] = arr[j+1];

arr[j+1] = temp;

}

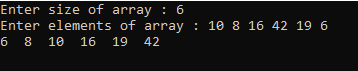
for(i=0;i<n;i++)

printf("%d ",arr[i]);

printf("\n");

}

***OUTPUT :***



***b. Selection sort.***

#include<stdio.h>

#include<stdlib.h>

#define SIZE 100

void selection\_sort(int \*arr,int n);

int main()

{

int arr[SIZE];

printf("Enter size of array : ");

int n;

scanf("%d",&n);

printf("Enter elements of array : ");

int i;

for(i=0;i<n;i++)

scanf("%d",&arr[i]);

selection\_sort(arr,n);

return 0;

}

void selection\_sort(int \*arr,int n)

{

int i,j,temp,k;

for(i=0;i<n;i++)

{

k = i;

for(j=i+1;j<n;j++)

if(arr[j]<arr[k])

k = j;

temp = arr[i];

arr[i] = arr[k];

arr[k] = temp;

}

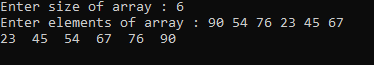
for(i=0;i<n;i++)

printf("%d ",arr[i]);

printf("\n");

}

***OUTPUT :***



***c. Insertion sort.***

#include<stdio.h>

#include<stdlib.h>

#define SIZE 100

void insertion\_sort(int \*arr,int n);

int main()

{

int arr[SIZE];

printf("Enter size of array : ");

int n;

scanf("%d",&n);

printf("Enter elements of array : ");

int i;

for(i=0;i<n;i++)

scanf("%d",&arr[i]);

insertion\_sort(arr,n);

return 0;

}

void insertion\_sort(int \*arr,int n)

{

int i,j,temp;

for(i=1;i<n;i++)

{

temp = arr[i];

for(j=i-1;j>=0 && temp<arr[j];j--)

arr[j+1] = arr[j];

arr[j+1] = temp;

}

for(i=0;i<n;i++)

printf("%d ",arr[i]);

printf("\n");

}

***OUTPUT :***



***d. Merge sort***

#include<stdio.h>

void merge(int arr[],int l,int k,int r)

{

int i=l,j=k+1;

int res[r-l+1];

int top=0;

while(i<=k && j<=r)

{

if(arr[i]>arr[j])

{

res[top]=arr[j];

j++;

top++;

}

else

{

res[top] = arr[i];

i++;

top++;

}

}

if(i<=k)

{

while(i<=k)

{

res[top]=arr[i];

top++;

i++;

}

}

if(j<=r)

{

while(j<=r)

{

res[top]=arr[j];

top++;

j++;

}

}

for(i=0;i<top;i++)

arr[l+i] = res[i];

}

void mergesort(int arr[],int l,int r)

{

if(l>=r)

return;

int k = (l+r)/2;

mergesort(arr,l,k);

mergesort(arr,k+1,r);

merge(arr,l,k,r);

}

int main()

{

int n;

printf("Enter size of array : ");

scanf("%d",&n);

int i,arr[n];

for(i=0;i<n;i++)

scanf("%d",&arr[i]);

mergesort(arr,0,n-1);

for(i=0;i<n;i++)

printf("%d ",arr[i]);

return 0;

}

***OUTPUT :***



***e. Quick sort.***

#include<stdio.h>

int partition(int arr[],int l,int u)

{

int i=l,j=u+1,temp;

do{

do{

i++;

}while(i<=u && arr[i]<arr[l]);

do{

j--;

}while(arr[j]>arr[l]);

if(j>i)

{

temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

}

}while(j>i);

temp = arr[l];

arr[l] = arr[j];

arr[j] = temp;

return j;

}

void quick\_sort(int arr[],int l,int u)

{

if(l>=u)

return;

int k = partition(arr,l,u);

quick\_sort(arr,l,k-1);

quick\_sort(arr,k+1,u);

}

int main()

{

int n;

printf("Enter size of array : ");

scanf("%d",&n);

int i,arr[n];

for(i=0;i<n;i++)

scanf("%d",&arr[i]);

quick\_sort(arr,0,n-1);

for(i=0;i<n;i++)

printf("%d ",arr[i]);

return 0;

}

***OUTPUT :***



***f. Heap sort.***

#include<stdio.h>

int n;

void build\_max\_heap(int arr[]);

void max\_heapify(int arr[],int i);

void heap\_sort(int arr[]);

int main()

{

int h;

printf("Enter number of nodes in array : ");

scanf("%d",&h);

n = h;

int arr[h+1];

int i;

for(i=1;i<=h;i++)

scanf("%d",&arr[i]);

heap\_sort(arr);

for(i=1;i<=h;i++)

printf("%d ",arr[i]);

return 0;

}

void heap\_sort(int arr[])

{

build\_max\_heap(arr);

while(n>=2)

{

int temp = arr[1];

arr[1] = arr[n];

arr[n] = temp;

n--;

max\_heapify(arr,1);

}

}

void build\_max\_heap(int arr[])

{

int i = n/2 + n%2;

while(i>=1)

{

max\_heapify(arr,i);

i--;

}

}

void max\_heapify(int arr[],int i)

{

int l = 2\*i;

int r = l+1;

int largest = i;

if(l<=n && arr[l]>arr[largest])

largest = l;

if(r<=n && arr[r]>arr[largest])

largest = r;

if(largest!=i)

{

int temp = arr[i];

arr[i] = arr[largest];

arr[largest] = temp;

max\_heapify(arr,largest);

}

}

***OUTPUT :***



***g. Shell sort.***

#include<stdio.h>

#define SIZE 100

void shell\_sort(int arr[],int n);

int main()

{

int n;

printf("Enter number of elements : ");

scanf("%d",&n);

int i,arr[SIZE];

for(i=0;i<n;i++)

scanf("%d",&arr[i]);

shell\_sort(arr,n);

for(i=0;i<n;i++)

printf("%d ",arr[i]);

return 0;

}

void shell\_sort(int arr[],int n)

{

int g,i,j;

for(g=n/2;g>0;g/=2)

{

for(i=g;i<n;i++)

{

int temp = arr[i];

for(j=i;j>=g && arr[j-g]>temp;j-=g)

arr[j] = arr[j-g];

arr[j] = temp

;

}

}

}

***OUTPUT :***



***h. Cook- Kim sort.***

#include<stdio.h>

#define SIZE 100

int arr[SIZE];

void insertion\_sort(int arr[],int n);

void cook\_kim\_sort(int arr1[],int arr2[],int i,int j);

int main()

{

int n;

scanf("%d",&n);

int i,arr1[SIZE],arr2[SIZE];

for(i=0;i<n;i++)

scanf("%d",&arr[i]);

int j=0,k=0;

for(i=0;i<n;i++)

{

if(i!=n-1 && arr[i]>arr[i+1])

{

arr1[j++] = arr[i];

arr1[j++] = arr[++i];

}

else

{

if(arr[i]>arr[k-1])

arr2[k++] = arr[i];

else

arr1[j++] = arr[i];

}

}

cook\_kim\_sort(arr1,arr2,j,k);

for(i=0;i<n;i++)

printf("%d ",arr[i]);

return 0;

}

void insertion\_sort(int arr[],int n)

{

int i,j;

for(i=0;i<n;i++)

{

int temp = arr[i];

for(j=i-1;j>=0 && arr[j]>temp;j--)

arr[j+1] = arr[j];

arr[j+1] = temp;

}

}

void cook\_kim\_sort(int arr1[],int arr2[],int i,int j)

{

insertion\_sort(arr1,i);

int x=0,y=0,z=0;

while(x<i && y<j)

{

if(arr1[x]>arr2[y])

arr[z++] = arr2[y++];

else

arr[z++] = arr1[x++];

}

if(x!=i)

{

while(x!=i)

arr[z++] = arr1[x++];

}

if(y!=j)

{

while(y!=j)

arr[z++] = arr2[y++];

}

}

***OUTPUT:***



***i. Topological sort.***

#include<stdio.h>

#include<stdlib.h>

struct graph

{

int count;

struct graph \*next;

};

typedef struct graph Graph;

void topological\_sort(Graph \*start,int n);

int main()

{

Graph \*start,\*p;

int n;

printf("Enter number of nodes in graph : ");

scanf("%d",&n);

start = (Graph \*)malloc((n+1)\*sizeof(Graph));

int i,x,y;

for(i=0;i<=n;i++)

{

start[i].count = 0;

start[i].next = NULL;

}

printf("Enter Edges (V1 V2, means V1--->V2) of Acyclic Directed Graph:\n");

printf("-1 to stop\n");

while(1)

{

scanf("%d%d",&x,&y);

if(x==-1)

break;

Graph \*temp = (Graph \*)malloc(sizeof(Graph));

temp->count = y;

temp->next = start[x].next;

start[x].next = temp;

(start[y].count)++;

}

printf("Topological sort is :\n");

topological\_sort(start,n);

free(start);

return 0;

}

void topological\_sort(Graph \*start,int n)

{

Graph \*p=start;

while(1)

{

p = start;

int i;

for(i=1;i<=n;i++)

if(start[i].count==0)

break;

if(i==n+1)

break;

start[i].count=-1;

printf("%d ",i);

p = start[i].next;

while(p!=NULL)

{

(start[p->count].count)--;

p = p->next;

}

}

}

***OUTPUT :***

