**Roll No**EN19CS3T1001



Knowledge is power

**Medi-Caps University**

**Indore**

**Department of** Computer Science Engineering

**Name** Mr. Ammar SusnerWala

**Class** CSE  **Section** E

**Subject** DESIGN AND ANALYSIS OF ALGORITHM

**Session** JAN-MAY 2021 **Semester** VI(even)

**DAA Practical File**

|  |  |  |
| --- | --- | --- |
| **S.No.** | **Practical Name** | **Page No.** |
| **1.** | **Write a program to implement the Bubble Sort** | **1** |
| **2.** | **Write a program to implement the Insertion Sort** | **3** |
| **3.** | **Write a program to implement the Selection Sort** | **5** |
| **4.** | **Write a program to implement the Linear Search** | **7** |
| **5.** | **Write a program to implement the Binary Search** | **9** |
| **6.** | **Write a program to demonstrate the Quick Sort Algorithm** | **12** |
| **7.** | **Write a program to demonstrate the Merge Sort Algorithm** | **16** |
| **8.** | **Write a program to demonstrate the Heap Sort Algorithm** | **19** |
| **9.** | **Write a program to demonstrate the Radix Sort Algorithm** | **22** |
| **10** | **Write a program to demonstrate the Strassen’s Matrix Multiplication** | **25** |
| **11** | **Write a program to demonstrate the Binary Search with recursive method** | **26** |
| **12** | **Write a program to demonstrate the Optimal Merge Pattern** | **28** |
| **13** | **Write a program to demonstrate the Fractional Knapsack Problem** | **30** |
| **14** | **Write a program to demonstrate the dijkstras algorithm in c** | **32** |

**1.Write a program to implement the bubble sort**

**Code :-**#include <stdio.h>

int main()

{ int a[100],i,j,temp,n;

printf("Enter the size of array\n");

scanf("%d",&n);

printf("Enter the elements in array\n");

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

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

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

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

{ if(a[j]>a[j+1])

{ temp=a[j];

a[j]=a[j+1];

a[j+1]=temp; }

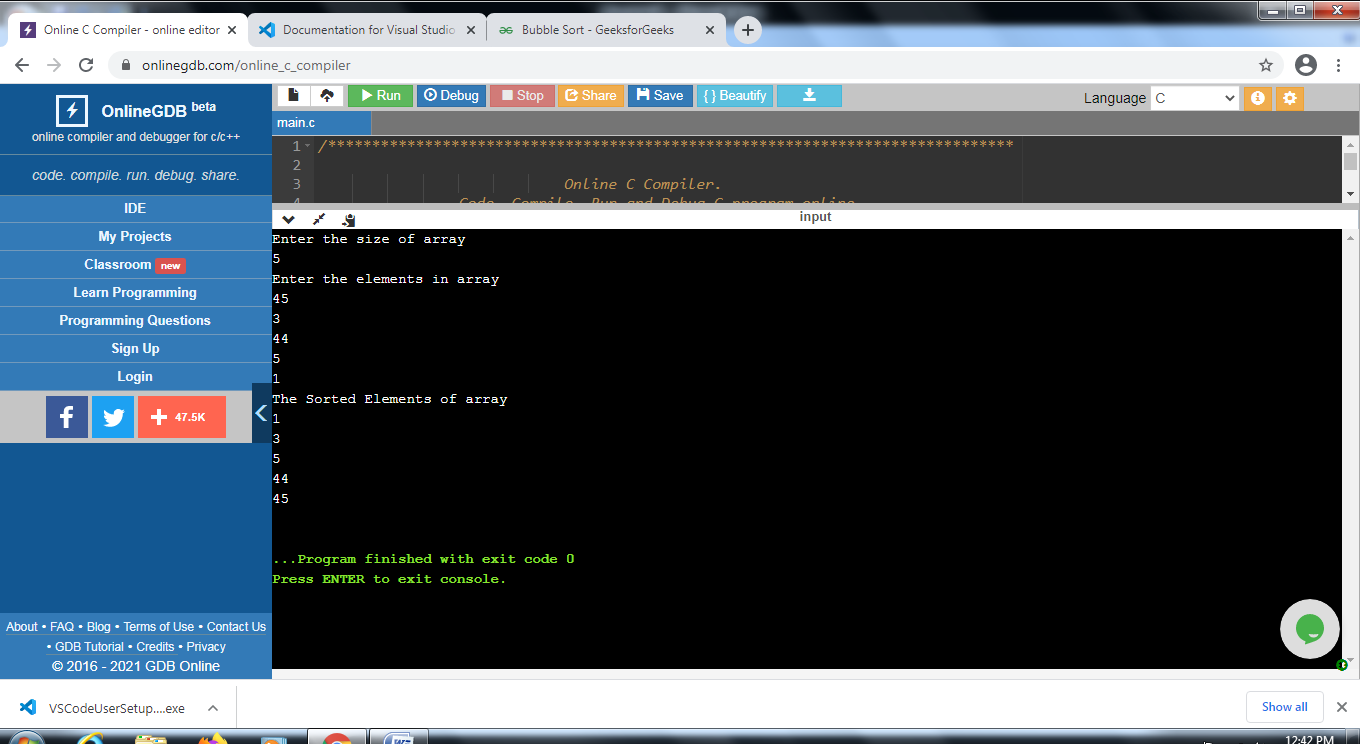
} }

printf("The Sorted Elements of array\n");

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

printf("%d\n",a[i]);

return 0;

**2.Write a program to implement the insertion sort**

**Code:-**#include<stdio.h>

int main()

{

int a[50],i,j,key,n;

printf("Enter the size of array\n");

scanf("%d",&n);

printf("Enter the elements in array\n");

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

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

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

{

key=a[i];

j=i-1;

while(j>=0&&a[j]>key)

{

a[j+1]=a[j];

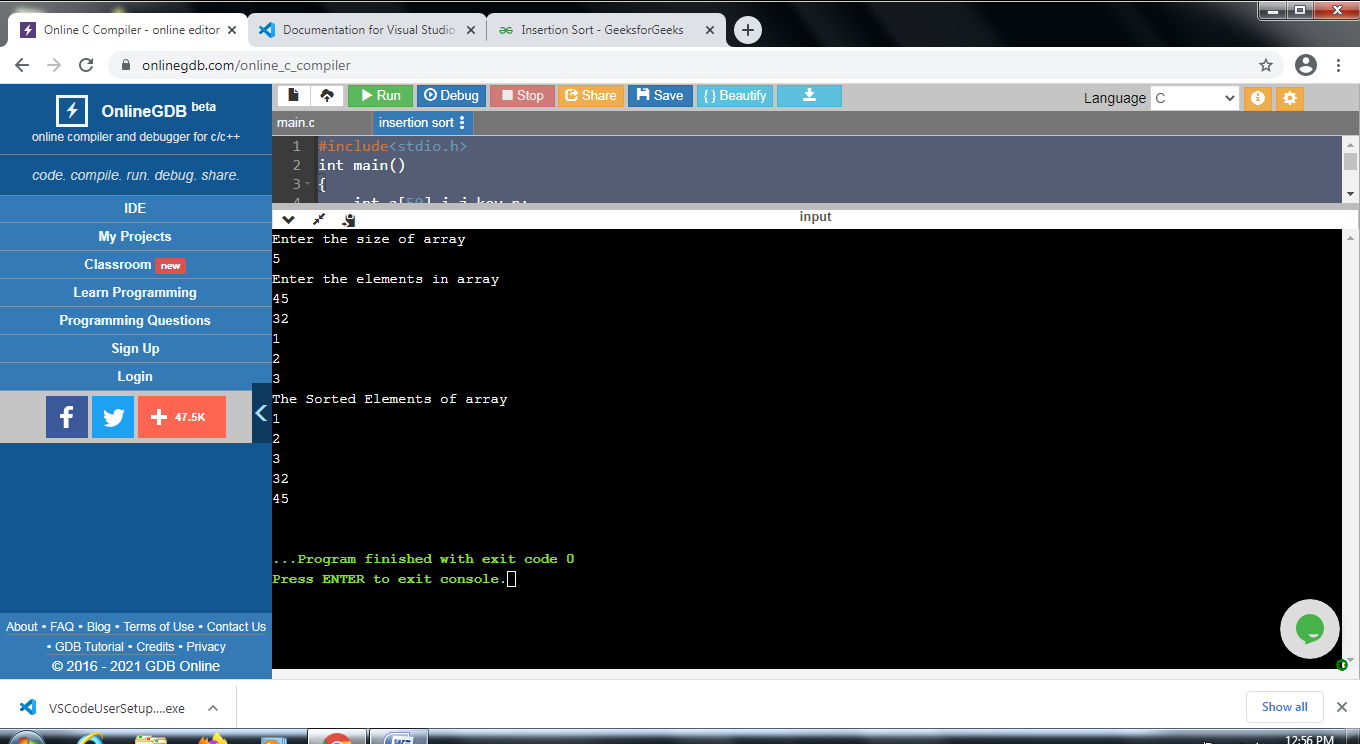
j=j-1;

}

a[j+1]=key; }

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

printf(“%d\n”.a[i]);

}

**3. Write a program to implement the Selection Sort**

**Code:-**

#include<stdio.h>

int main()

{ int a[100],i,j,n,temp,min;

printf("Enter the size of array\n");

scanf("%d",&n);

printf("Enter the elements in array\n");

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

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

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

{ min=i;

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

{ if(a[j]<a[min])

{ min=j;

temp=a[min];

a[min]=a[i];

a[i]=temp;

}

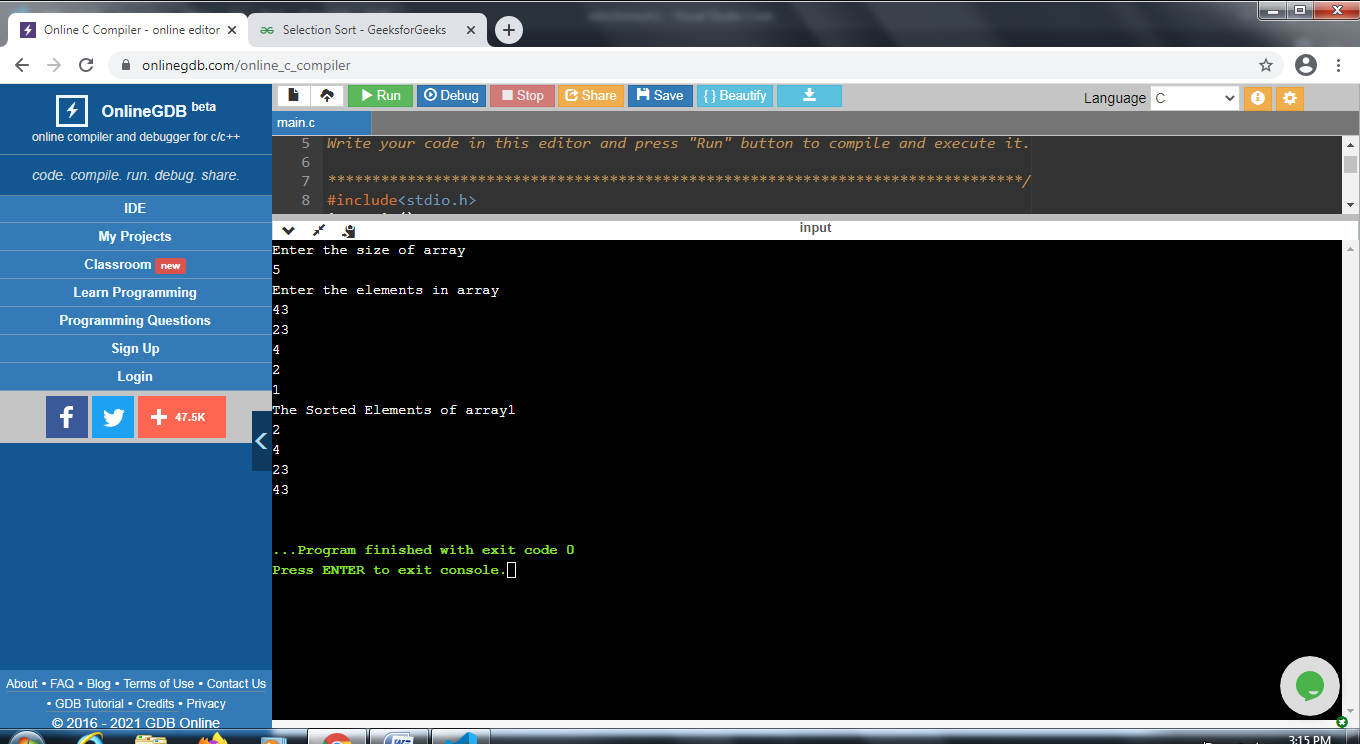
}

}

printf("The Sorted Elements of array");

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

printf("%d\n",a[i]);}



**4.Write a program to implement the Linear Search**

**Code:-**#include<stdio.h>

int main()

{ int a[100],i,n,search,flag,j;

printf("Enter the size of array\n");

scanf("%d",&n);

printf("Enter the elements in array\n");

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

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

printf("Enter the number for which you want to search\n");

scanf("%d",&search);

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

{ if(a[i]==search)

{ flag=1;

j=i;}

}

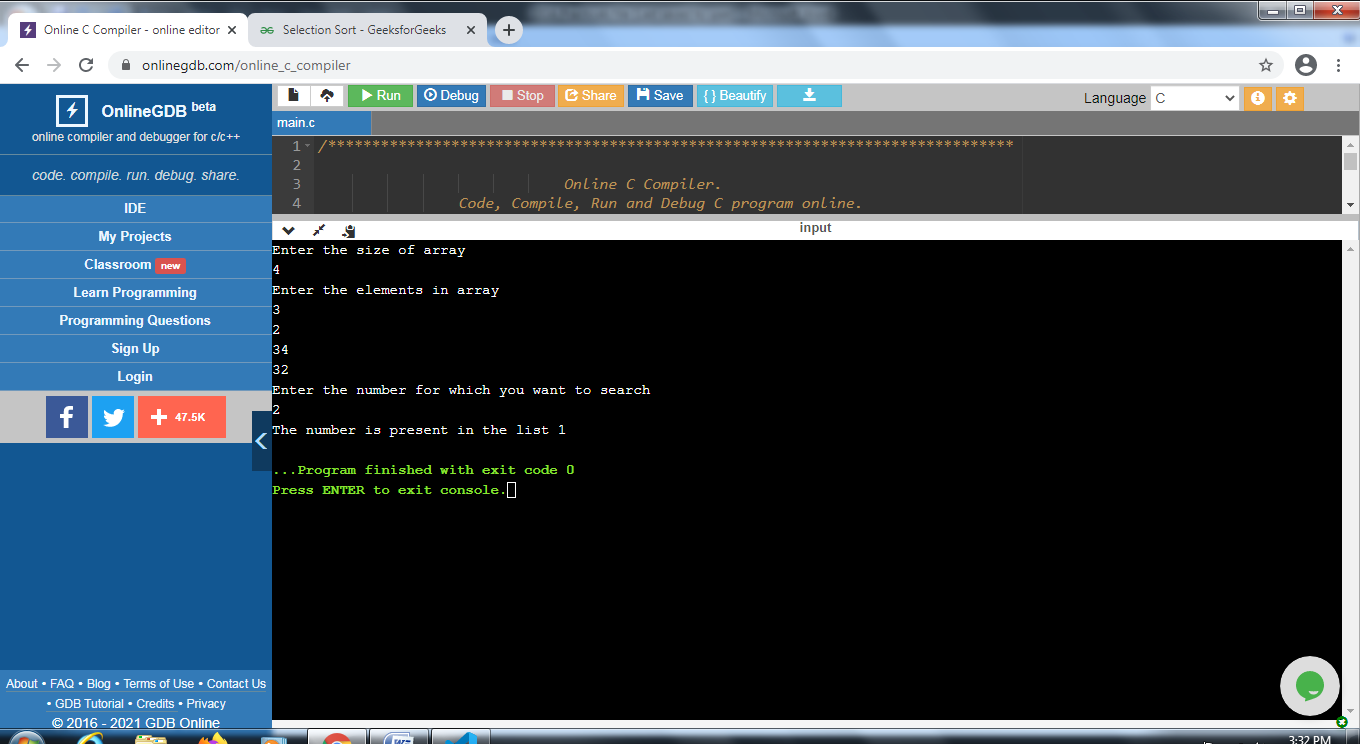
if(flag==1)

printf("The number is present in the list %d",j);

else

printf("Not Found");

}



**5. Write a program to implement the Binary Search**

**Code:-**#include<stdio.h>

int binarySearch(int[], int, int, int);

void main ()

{ int a[100],n,i;

printf("Enter the size of array\n");

scanf("%d",&n);

printf("Enter the elements in array\n");

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

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

int item, location=-1;

printf("Enter the item which you want to search ");

scanf("%d",&item);

location = binarySearch(a, 0, 9, item);

if(location != -1)

{ printf("Item found at location %d",location);

} else

{ printf("Item not found");

} }

int binarySearch(int a[], int beg, int end, int item)

{ int mid;

if(end >= beg)

{ mid = (beg + end)/2;

if(a[mid] == item)

{ return mid+1;

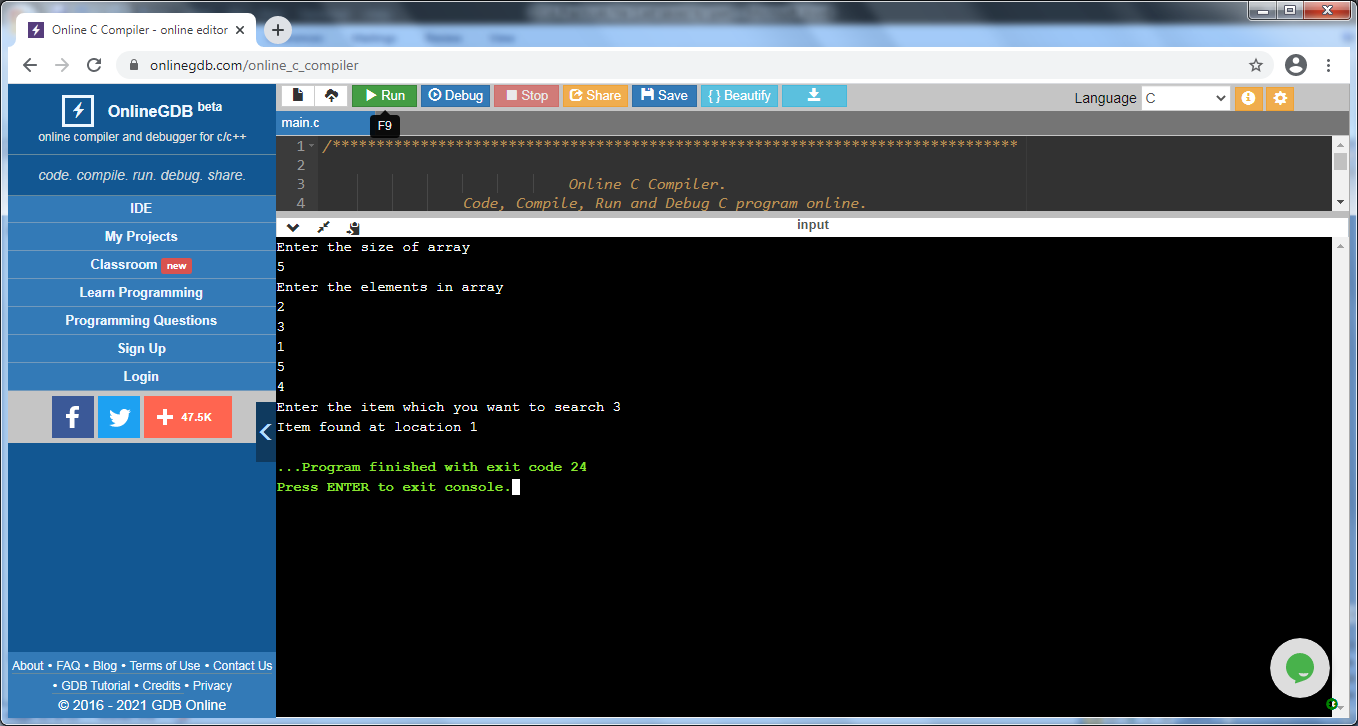
}else if(a[mid] < item)

{ return binarySearch(a,mid+1,end,item);

} else

{ return binarySearch(a,beg,mid-1,item);

} }}



6. Write to program to demonstrate the quick sort algorithm

Code:- #include <stdio.h>

int main()

{ int list[50];

int size, i;

printf("Enter the number of elements ");

scanf("%d", &size);

printf("Enter the elements of the array\n");

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

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

quicksort(list, 0, size - 1);

printf("The sorted array is\n");

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

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

printf("\n");

return 0;

}

void quicksort(int list[], int low, int high)

{ int pivot, i, j, temp;

if (low < high)

{ pivot = low;

i = low;

j = high;

while (i < j)

{ while (list[i] <= list[pivot] && i <= high)

{

i++;

}

while (list[j] > list[pivot] && j >= low)

{

j--;

}

if (i < j)

{ temp = list[i];

list[i] = list[j];

list[j] = temp;

}}

temp = list[j];

list[j] = list[pivot];

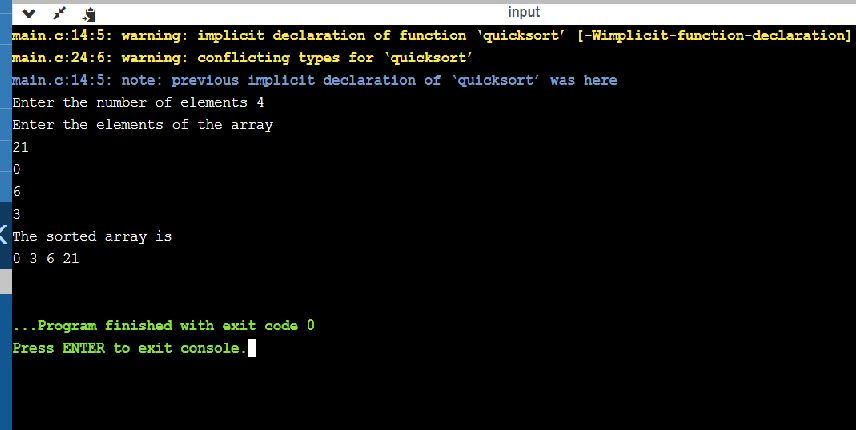
list[pivot] = temp;

quicksort(list, low, j - 1);

quicksort(list, j + 1, high);

}

}



7. Write a program to demonstrate the merge sort algorithm

Code:- #include <stdio.h>

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

{ int i, j, k;

int n1 = m - l + 1;

int n2 = r - m;

int L[n1], R[n2];

for (i = 0; i < n1; i++)

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

for (j = 0; j < n2; j++)

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

i = 0;

j = 0;

k = l;

while (i < n1 && j < n2) {

if (L[i] <= R[j]) {

arr[k] = L[i];

i++;

}

else {

arr[k] = R[j];

j++;

} k++;

}

while (i < n1) {

arr[k] = L[i];

i++;

k++;

}

while (j < n2) {

arr[k] = R[j];

j++;

k++;

}

}

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

{

if (l < r) {

int m = l + (r - l) / 2;

mergeSort(arr, l, m);

mergeSort(arr, m + 1, r);

merge(arr, l, m, r);

}

}

int main()

{

int arr[50],n,i;

printf("Enter the size of array\n");

scanf("%d",&n);

printf("Enter the elements in array\n");

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

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

mergeSort(arr, 0,n-1);

printf("\nSorted array is \n");

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

{

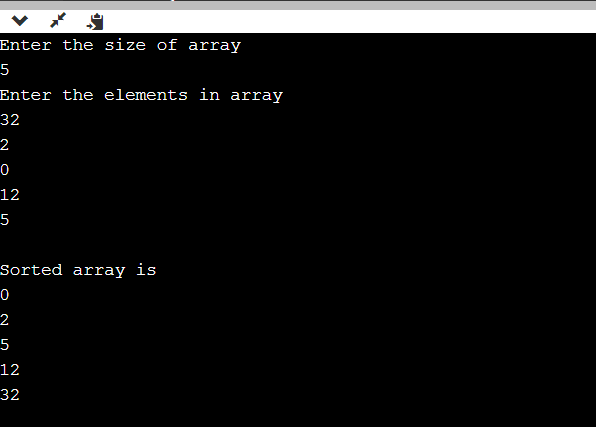
printf("%d",arr[i]);

printf("\n");

}

return 0;

}



8. Write a program to demonstrate the Heap Sort Algorithm

Code: - #include <stdio.h>

void heapify (int a[], int n, int i) {

int largest = i;

int left = 2 \* i + 1;

int right = 2 \* i + 2;

if (left < n && a[left] > a[largest])

largest = left;

if (right < n && a[right] > a[largest])

largest = right;

if (largest != i) {

int temp= a[i];

a[i]=a[largest];

a[largest]=temp;

heapify(a, n, largest);

} }

void heapSort(int a[], int n) {

int i;

for ( i = (n / 2 )-1 ; i >= 0; i--)

heapify(a, n, i);

for ( i = n - 1; i >= 0; i--)

{ int temp = a[0];

a[0]=a[i];

a[i]=temp;

heapify(a, i, 0);

} }

int main() {

int a[50],n,i;

printf("Enter the size of Array\n");

scanf("%d",&n);

printf("Enter the elements in Array\n");

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

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

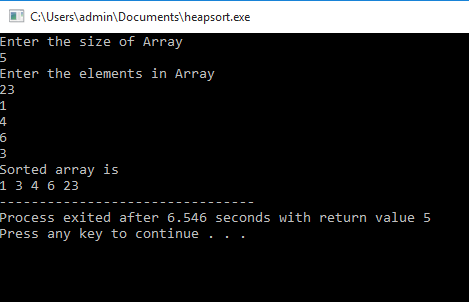
heapSort(a, n);

printf("Sorted array is \n");

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

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

}



9. Write a program to demonstrate the Radix Sort Algorithm

Code: -#include <stdio.h>

int getMax(int array[], int n) {

int max = array[0];

int i;

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

if (array[i] > max)

max = array[i];

return max;

}

void countingSort(int array[], int size, int place) {

int output[size + 1];

int max = (array[0] / place) % 10;

int i;

for (i = 1; i < size; i++) {

if (((array[i] / place) % 10) > max)

max = array[i];

}

int count[max + 1];

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

count[i] = 0;

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

count[(array[i] / place) % 10]++;

for (i = 1; i < 10; i++)

count[i] += count[i - 1];

for (i = size - 1; i >= 0; i--) {

output[count[(array[i] / place) % 10] - 1] = array[i];

count[(array[i] / place) % 10]--;

}

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

array[i] = output[i];

}

void radixsort(int array[], int size) {

int max = getMax(array, size);

int place;

for ( place = 1; max / place > 0; place \*= 10)

countingSort(array, size, place);

}

int main() {

int a[50],size,i;

printf("Enter the size of Array\n");

scanf("%d",&size);

printf("Enter the Elements in Array\n");

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

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

radixsort(a, size);

printf("The Sorted Elements of Array\n");

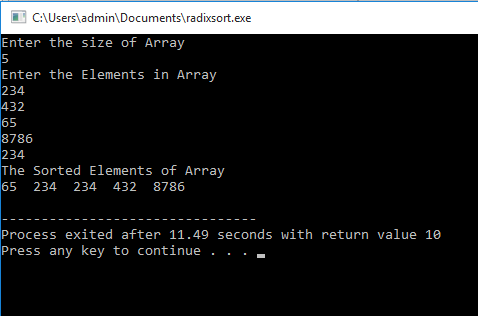
for (i = 0; i < size; ++i) {

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

}

printf("\n");;

}



10. Write a program to demonstrate the Strassen’s Matrix Multiplication

Code: - #include<stdio.h>

int main()

{ int a[2][2],b[2][2],i,j;

int c[2][2];

printf("Enter the elements in First Matrix\n");

for(i=0;i<2;i++)

{ for(j=0;j<2;j++)

{ scanf("%d",&a[i][j]);

}

}

printf("Enter the Elements in Second Matrix\n");

for(i=0;i<2;i++)

{ for(j=0;j<2;j++)

{ scanf("%d",&b[i][j]);

}

}

int p1=(a[0][0]+a[1][1])\*(b[0][0]+b[1][1]);

int p2=(a[1][0]+a[1][1])\*b[0][0];

int p3=a[0][0]\*(b[0][1]-b[1][1]);

int p4=a[1][1]\*(b[1][0]-b[0][0]);

int p5=(a[0][0]+a[0][1])\*b[1][1];

int p6=(a[1][0]-a[0][0])\*(b[0][0]+b[0][1]);

int p7=(a[0][1]-a[1][1])\*(b[1][0]+b[1][1]);

c[0][0]=(p1+p4)-(p5+p7);

c[0][1]=p3+p5;

c[1][0]=p2+p4;

c[1][1]=(p1+p3)-(p2+p6);

printf("The result of Stressen's Matrix Multiplication\n");

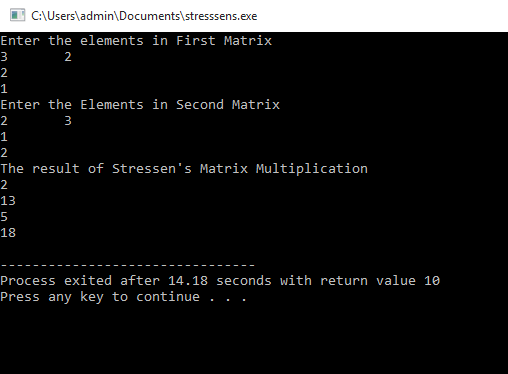
for(i=0;i<2;i++)

{ for(j=0;j<2;j++)

{ printf("%d",c[i][j]);

printf("\n");

} }}



11. Write a program to demonstrate the Binary Search Algorithm with recursive method

Code: -#include <stdio.h>

int main()

{ int a[50],i,n,searchelement,flag;

printf("Enter the size of Array\n");

scanf("%d",&n);

int mid=n/2;

printf("Enter the Elements in Array\n");

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

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

printf("Enter the element for Serach");

scanf("%d",&searchelement);

int result=binarySearch(a,0,n-1,searchelement);

if( result==-1)

printf("The Element is Not Found");

else

printf("Element is found%d", result);

}

int binarySearch(int a[],int start,int last,int searchelement)

{ if(last>=start)

{int mid=start+(last-1)/2;

if(a[mid]==searchelement)

return mid;

if(a[mid]>searchelement)

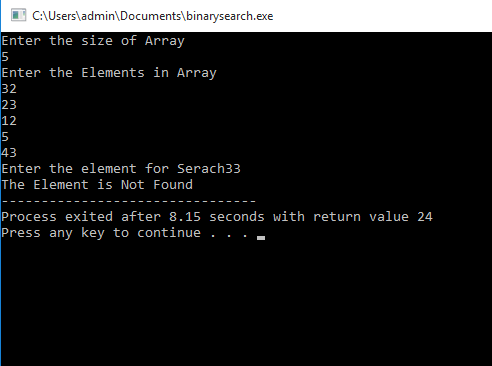
return binarySearch(a,start,mid-1,searchelement);

else

return binarySearch(a,mid+1,last,searchelement);

} return -1;

}

12. **Write** **a program to demonstrate the Optimal Merge Pattern**

**Code: -** #include<stdio.h>

int main()

{ int a[50],n,i,k,l;

int c[50];

printf("Enter the Size of Array\n");

scanf("%d",&n);

printf("Enter the Sorted Elements for optimal merge pattern\n");

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

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

i=0;

k=0;

c[k]=a[i]+a[i+1];

i=2;

while(i<n)

{ k++;

if((c[k-1]+a[i])<=(a[i]+a[i+1]))

{ c[k]=c[k-1]+a[i];

}

else

{c[k]=a[i]+a[i+1];

i=i+2;

while(i<n)

{ k++;

if((c[k-1]+a[i])<=c[k-2]+a[i])

{ c[k]=c[k-1]+a[i];

}

else

{ c[k]=c[k-2]+a[i];

}

i++;

}}

i++;

}

k++;

c[k]=c[k-1]+c[k-2];

printf("The Optimal Sum\n");

l=0;

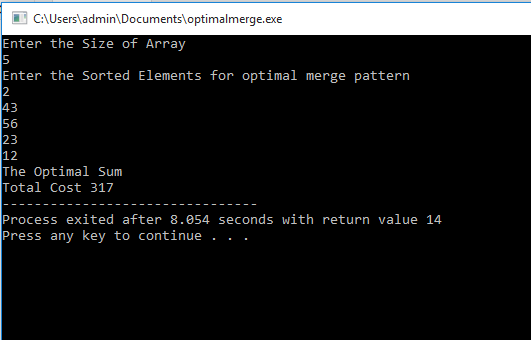
for(k=0;k<n-1;k++)

{l=l+c[k];

}

printf("Total Cost %d",l);

}



**13.Write a program to demonstrate the Fractional Knapsack Problem**

**Code: -** # include<stdio.h>

void knapsack(int n, float weight[], float profit[], float capacity) {

float x[20], tp = 0;

int i, j, u;

u = capacity;

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

x[i] = 0.0;

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

if (weight[i] > u)

break;

else {

x[i] = 1.0;

tp = tp + profit[i];

u = u - weight[i];

} }

if (i < n)

x[i] = u / weight[i];

tp = tp + (x[i] \* profit[i]);

printf("The result vector is \n ");

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

printf("%f\t", x[i]);

printf("\nMaximum profit is %f\n", tp);

}

int main()

{ float weight[20], profit[20], capacity;

int num, i, j;

float ratio[20], temp;

printf("Enter the no. of objects\n ");

scanf("%d", &num);

printf("Enter the Weights and profits of each object\n ");

for (i = 0; i < num; i++) {

scanf("%f %f", &weight[i], &profit[i]);

}

printf("\nEnter the capacityacity of knapsack ");

scanf("%f", &capacity);

for (i = 0; i < num; i++) {

ratio[i] = profit[i] / weight[i];

}

for (i = 0; i < num; i++) {

for (j = i + 1; j < num; j++) {

if (ratio[i] < ratio[j]) {

temp = ratio[j];

ratio[j] = ratio[i];

ratio[i] = temp;

temp = weight[j];

weight[j] = weight[i];

weight[i] = temp;

temp = profit[j];

profit[j] = profit[i];

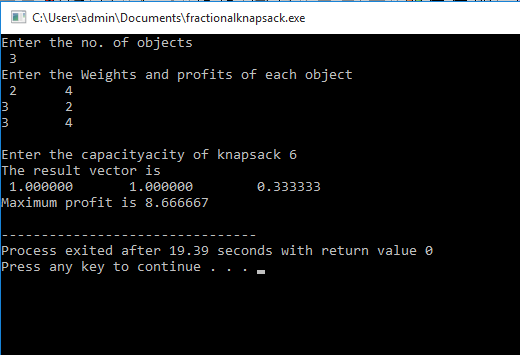
profit[i] = temp;

}}}

knapsack(num, weight, profit, capacity);

return 0;

}



**14.Write a program to demonstrate the dijkstras algorithm in c**

**Code:-** #include<stdio.h>

#include<conio.h>

#define INFINITY 9999

#define MAX 10

int main()

{

int G[MAX][MAX],i,j,n,u;

printf("Enter no. of vertices:");

scanf("%d",&n);

printf("\nEnter the adjacency matrix:\n");

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

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

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

printf("\nEnter the starting node:");

scanf("%d",&u);

dijkstra(G,n,u);

return 0;

}

void dijkstra(int G[MAX][MAX],int n,int startnode)

{ int cost[MAX][MAX],distance[MAX],pred[MAX];

int visited[MAX],count,mindistance,nextnode,i,j;

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

{ for(j=0;j<n;j++)

if(G[i][j]==0)

cost[i][j]=INFINITY;

else

cost[i][j]=G[i][j];

}

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

{ distance[i]=cost[startnode][i];

pred[i]=startnode;

visited[i]=0;

}

distance[startnode]=0;

visited[startnode]=1;

count=1;

while(count<n-1)

{ mindistance=INFINITY;

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

if(distance[i]<mindistance&&!visited[i])

{mindistance=distance[i];

nextnode=i;

}

visited[nextnode]=1;

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

if(!visited[i])

if(mindistance+cost[nextnode][i]<distance[i])

{distance[i]=mindistance+cost[nextnode][i];

pred[i]=nextnode; }

count++;

}

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

if(i!=startnode)

{printf("\nDistance of node%d=%d",i,distance[i]);

printf("\nPath=%d",i);

j=i;

do

{ j=pred[j];

printf("<-%d",j);

}

while(j!=startnode);

}}

