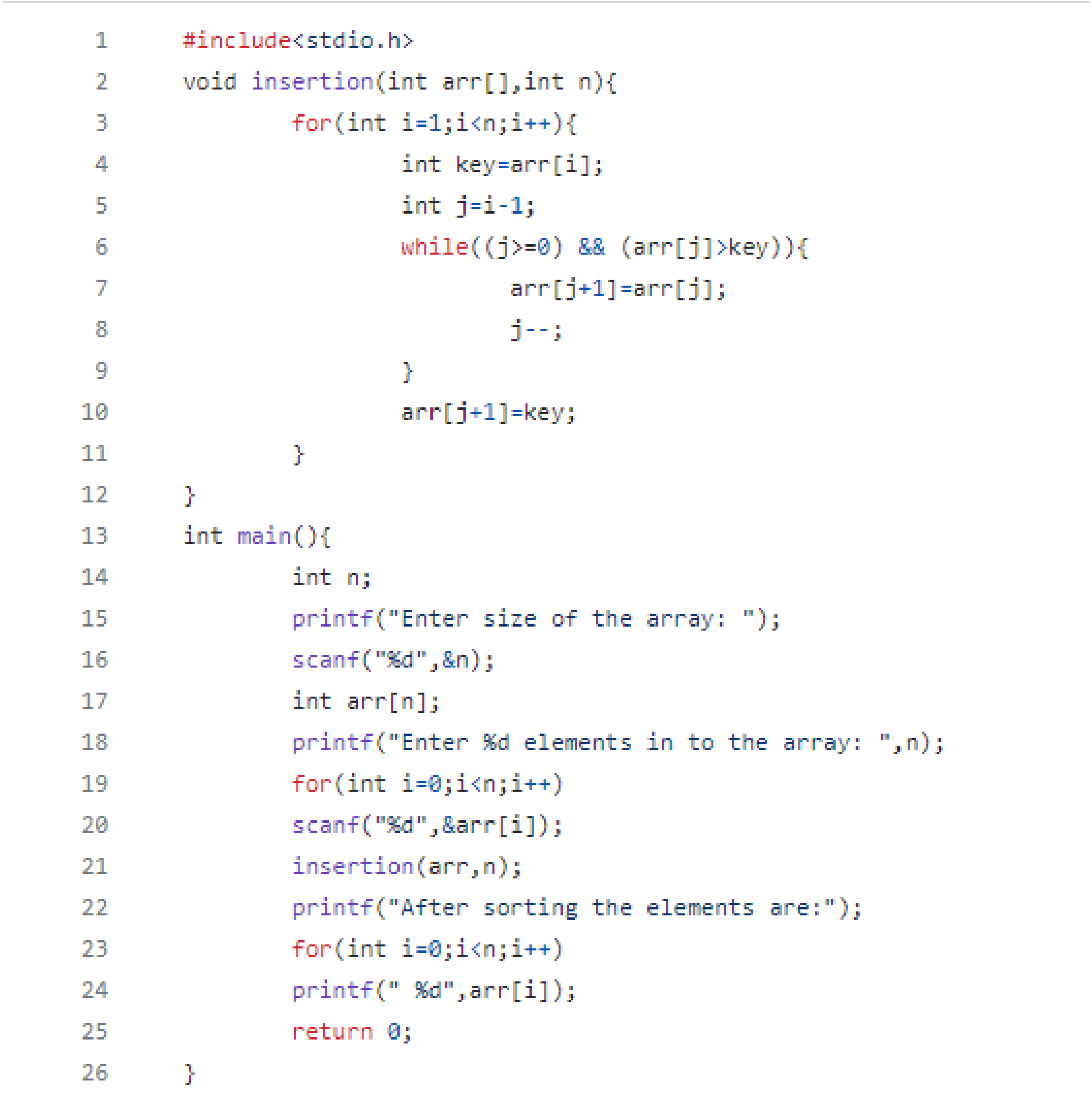
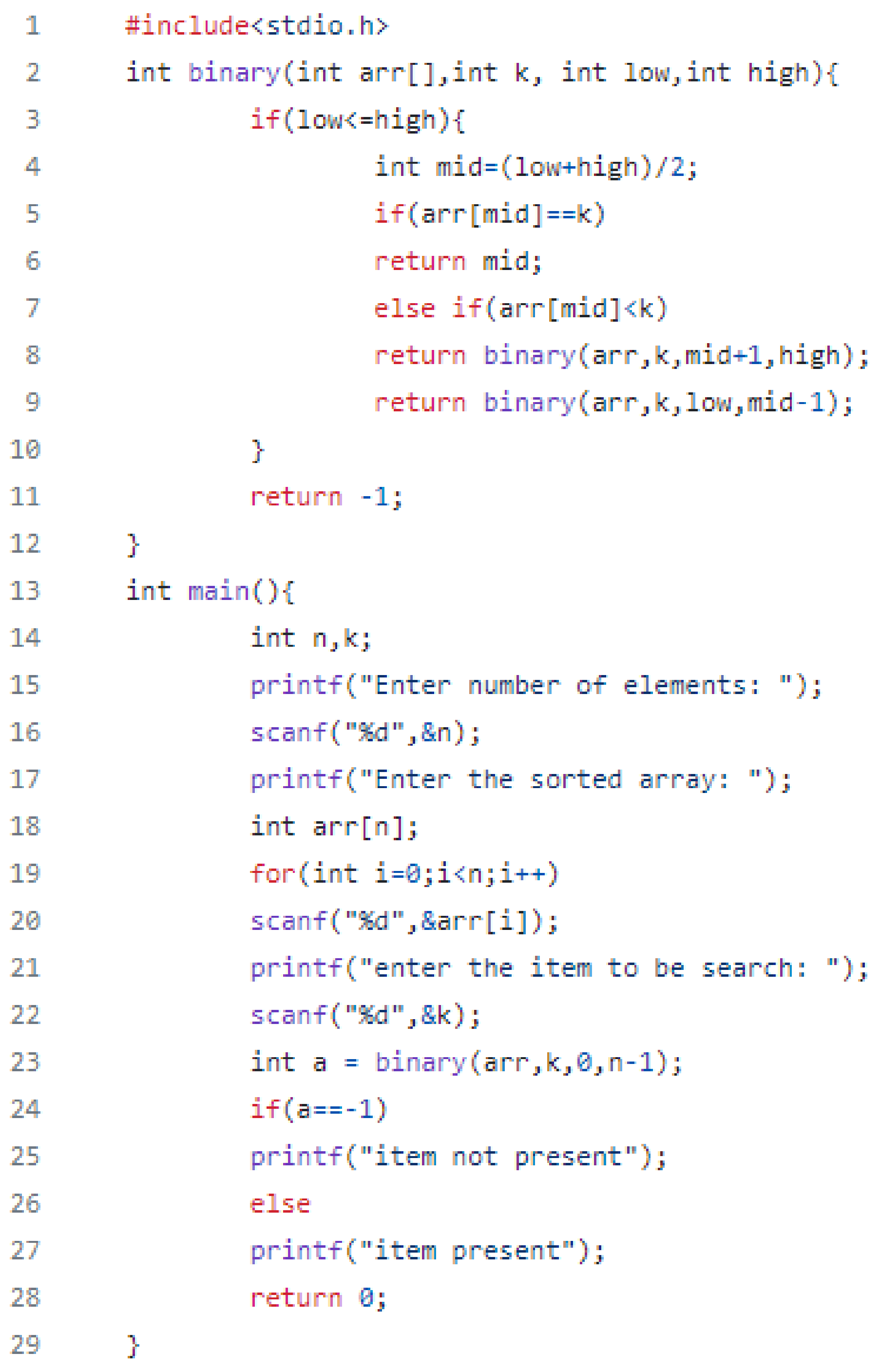
INSERTION SORT



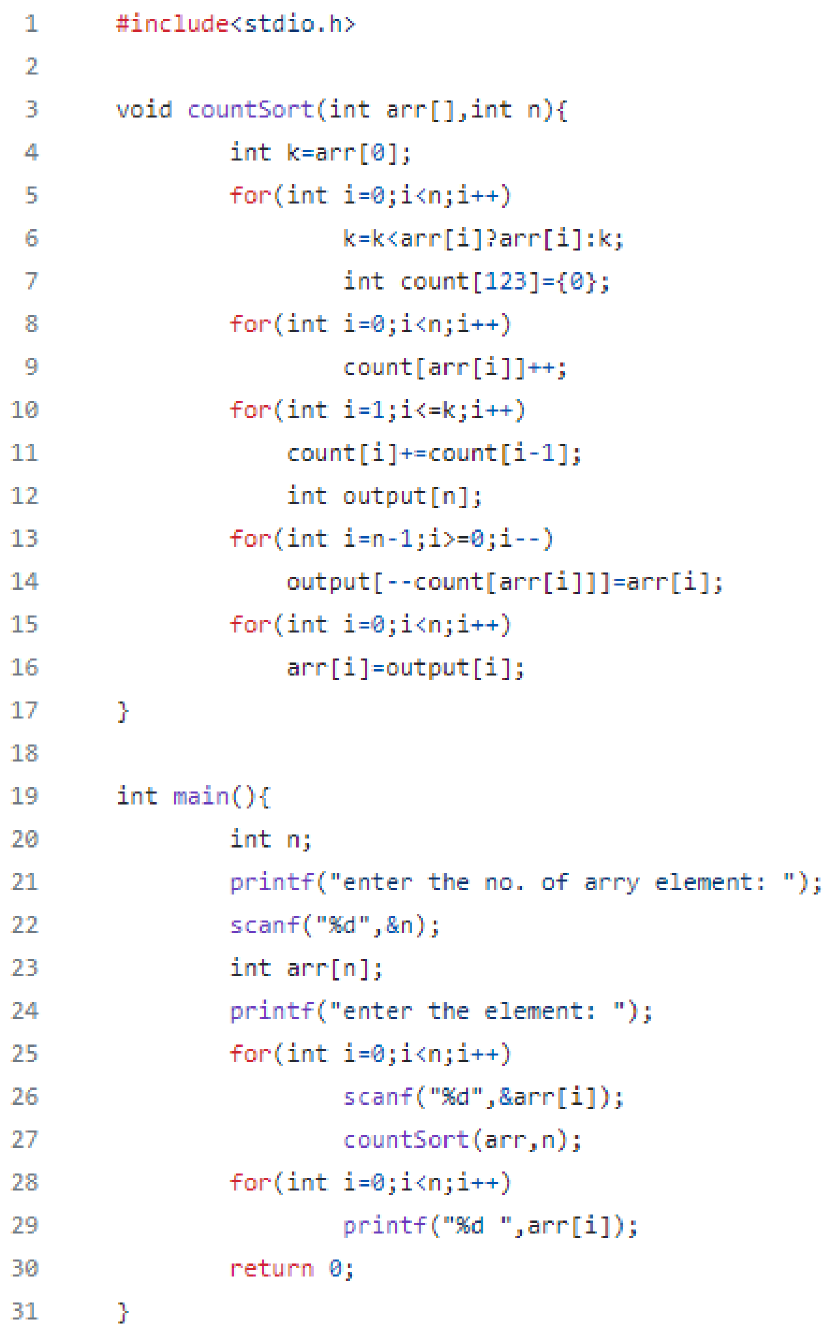
```c
#include<stdio.h>
void insertion(int arr[],int n){
    for(int i=1;i<n;i++){
        int key=arr[i];
        int j=i-1;
        while((j>=0) && (arr[j]>key)){
            arr[j+1]=arr[j];
            j--;
        }
        arr[j+1]=key;
    }
}
int main(){
    int n;
    printf("Enter size of the array: ");
    scanf("%d",&n);
    int arr[n];
    printf("Enter %d elements in to the array: ",n);
    for(int i=0;i<n;i++)
    scanf("%d",&arr[i]);
    insertion(arr,n);
    printf("After sorting the elements are:");
    for(int i=0;i<n;i++)
    printf(" %d",arr[i]);
    return 0;
}
```

BINARY SEARCH



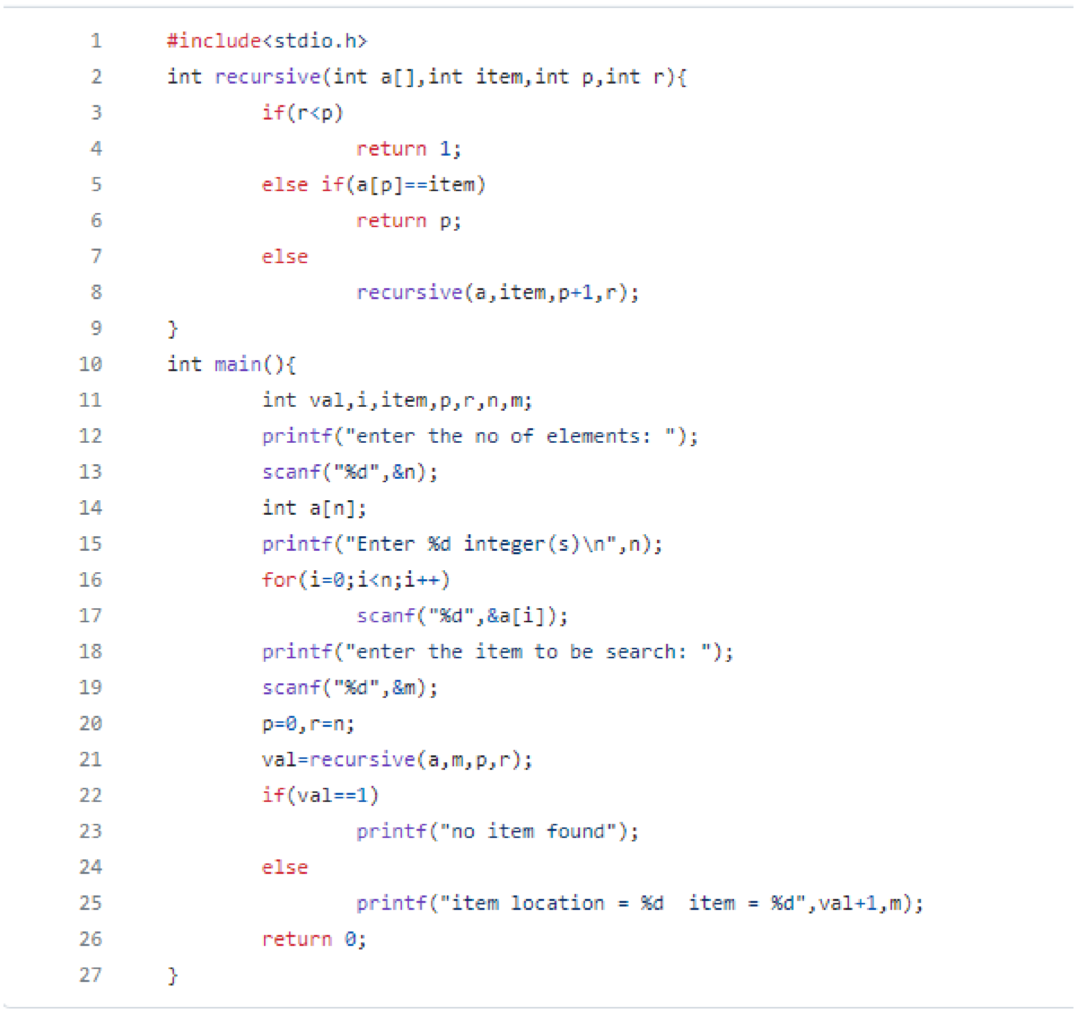
```c
#include<stdio.h>
int binary(int arr[],int k, int low,int high){
        if(low<=high){
                int mid=(low+high)/2;
                if(arr[mid]==k)
                return mid;
                else if(arr[mid]<k)
                return binary(arr,k,mid+1,high);
                return binary(arr,k,low,mid-1);
        }
        return -1;
}
int main(){
        int n,k;
        printf("Enter number of elements: ");
        scanf("%d",&n);
        printf("Enter the sorted array: ");
        int arr[n];
        for(int i=0;i<n;i++)
        scanf("%d",&arr[i]);
        printf("enter the item to be search: ");
        scanf("%d",&k);
        int a = binary(arr,k,0,n-1);
        if(a==-1)
        printf("item not present");
        else
        printf("item present");
        return 0;
}
```

COUNT SORT



```c
#include<stdio.h>

void countSort(int arr[],int n){
        int k=arr[0];
        for(int i=0;i<n;i++)
                k=k<arr[i]?arr[i]:k;
                int count[123]={0};
        for(int i=0;i<n;i++)
                count[arr[i]]++;
        for(int i=1;i<=k;i++)
            count[i]+=count[i-1];
            int output[n];
        for(int i=n-1;i>=0;i--)
            output[--count[arr[i]]]=arr[i];
        for(int i=0;i<n;i++)
            arr[i]=output[i];
}

int main(){
        int n;
        printf("enter the no. of arry element: ");
        scanf("%d",&n);
        int arr[n];
        printf("enter the element: ");
        for(int i=0;i<n;i++)
                scanf("%d",&arr[i]);
                countSort(arr,n);
        for(int i=0;i<n;i++)
                printf("%d ",arr[i]);
        return 0;
}
```

## LINEAR SEARCH



```c
#include<stdio.h>
int recursive(int a[],int item,int p,int r){
        if(r<p)
                return 1;
        else if(a[p]==item)
                return p;
        else
                recursive(a,item,p+1,r);
}
int main(){
        int val,i,item,p,r,n,m;
        printf("enter the no of elements: ");
        scanf("%d",&n);
        int a[n];
        printf("Enter %d integer(s)\n",n);
        for(i=0;i<n;i++)
                scanf("%d",&a[i]);
        printf("enter the item to be search: ");
        scanf("%d",&m);
        p=0,r=n;
        val=recursive(a,m,p,r);
        if(val==1)
                printf("no item found");
        else
                printf("item location = %d  item = %d",val+1,m);
        return 0;
}
```

SELECTION SORT

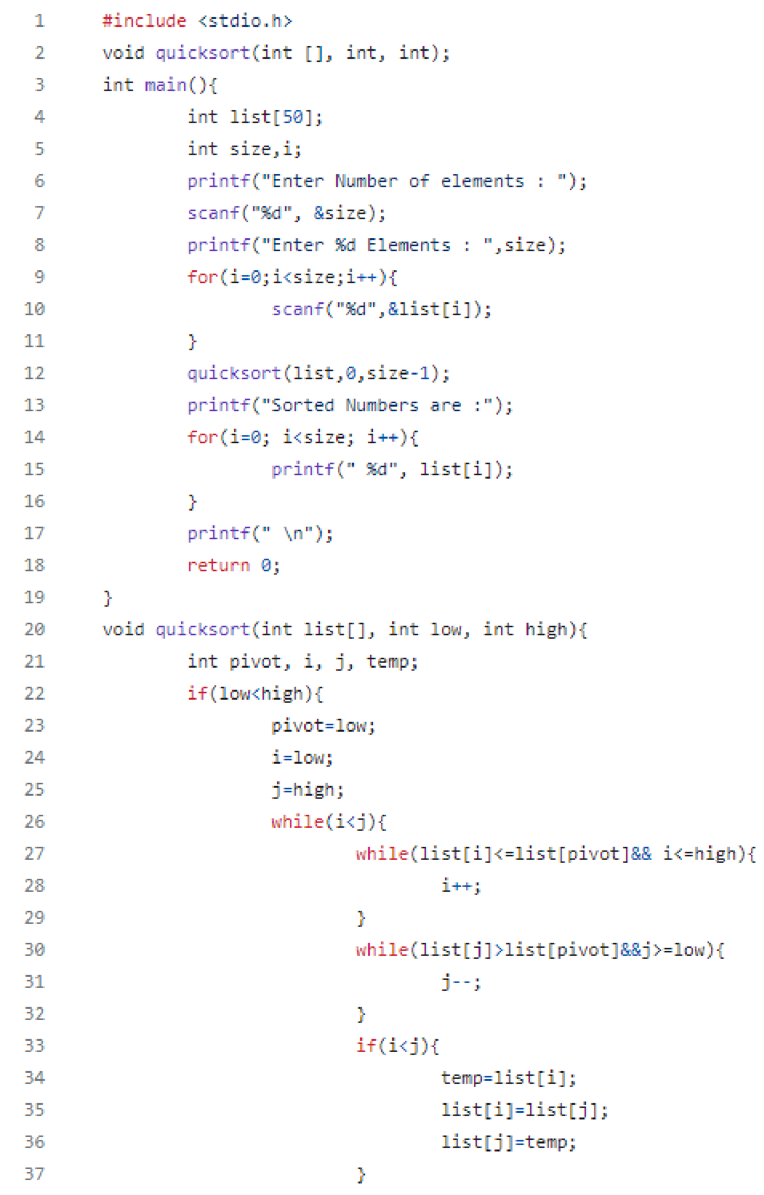
```c
#include<stdio.h>

void selection(int arr[],int n){

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

        {

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

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

                                arr[i]=arr[i]^arr[j];

                                arr[j]=arr[i]^arr[j];

                                arr[i]=arr[i]^arr[j];

                        }

                }

        }

}

int main(){

        int n;

        // scanf("%d",&n)

        printf("Enter size of the array : ");
```

```
            scanf("%d",&n);

            int arr[n];

            printf("Enter the elements :");

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

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

             selection(arr,n);

             printf("The sorted elements are : ");

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

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

             return 0;

    }
```

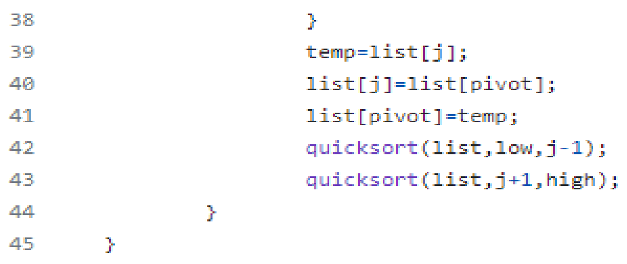
MERGE SORT

```c
#include<stdio.h>
#include<conio.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 merge_sort(int arr[],int l,int r)
{
        if(l<r){
                int m=(l+r)/2;
                merge_sort(arr,l,m);
                merge_sort(arr,m+1,r);
                merge(arr,l,m,r);
        }
}
void main(){
        int n,i,a[100];
        printf(" Enter How many Numbers : ");
        scanf("%d",&n);
        printf(" Enter %d Numbers :",n);
        for(i=0;i<n;i++){

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

        }
        merge_sort(a,0,n-1);
        printf(" Sorted Numbers are : ");
        for(i=0;i<n;i++)
        {

                printf("%d      ",a[i]); //use tab instead of space here

        }
}
```

QUICK SORT



```c
#include <stdio.h>
void quicksort(int [], int, int);
int main(){
        int list[50];
        int size,i;
        printf("Enter Number of elements : ");
        scanf("%d", &size);
        printf("Enter %d Elements : ",size);
        for(i=0;i<size;i++){
                scanf("%d",&list[i]);
        }
        quicksort(list,0,size-1);
        printf("Sorted Numbers are :");
        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;
                        }
```

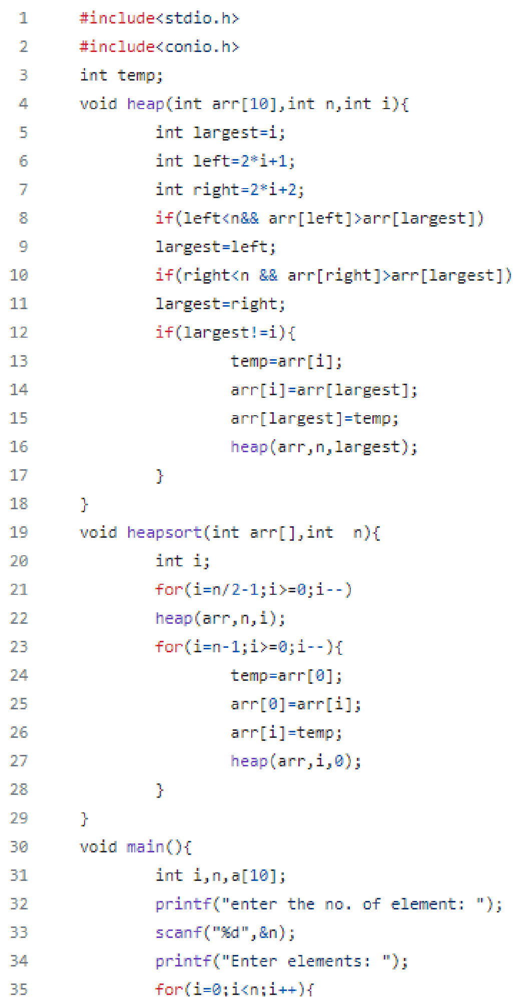


```c
                }
                temp=list[j];
                list[j]=list[pivot];
                list[pivot]=temp;
                quicksort(list,low,j-1);
                quicksort(list,j+1,high);
        }
}
```

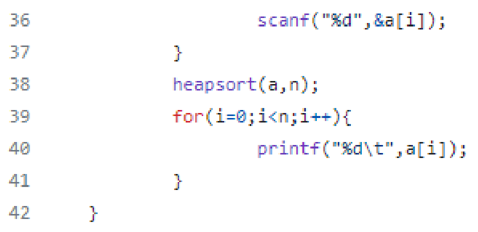
## DIVIDE AND CONQUER

```c
#include<stdio.h>
#include<conio.h>
void main(){
        int arr[10],n,i,max,min;
        printf("Enter the total number of Elements : ");
        scanf("%d",&n);
        printf("Enter the numbers : ");
        for(i=0;i<n;i++){
                scanf("%d",&arr[i]);
        }
        max=min=arr[0];
        for(i=0;i<n;i++){
                if(arr[i]>max){
                        max=arr[i];
                }
                if(arr[i]<min){
                        min=arr[i];
                }
        }
        printf("Minimum element in an array : %d",min);
        printf("\nMaximum element in an array : %d\n",max);
}
```

## Heap Sort

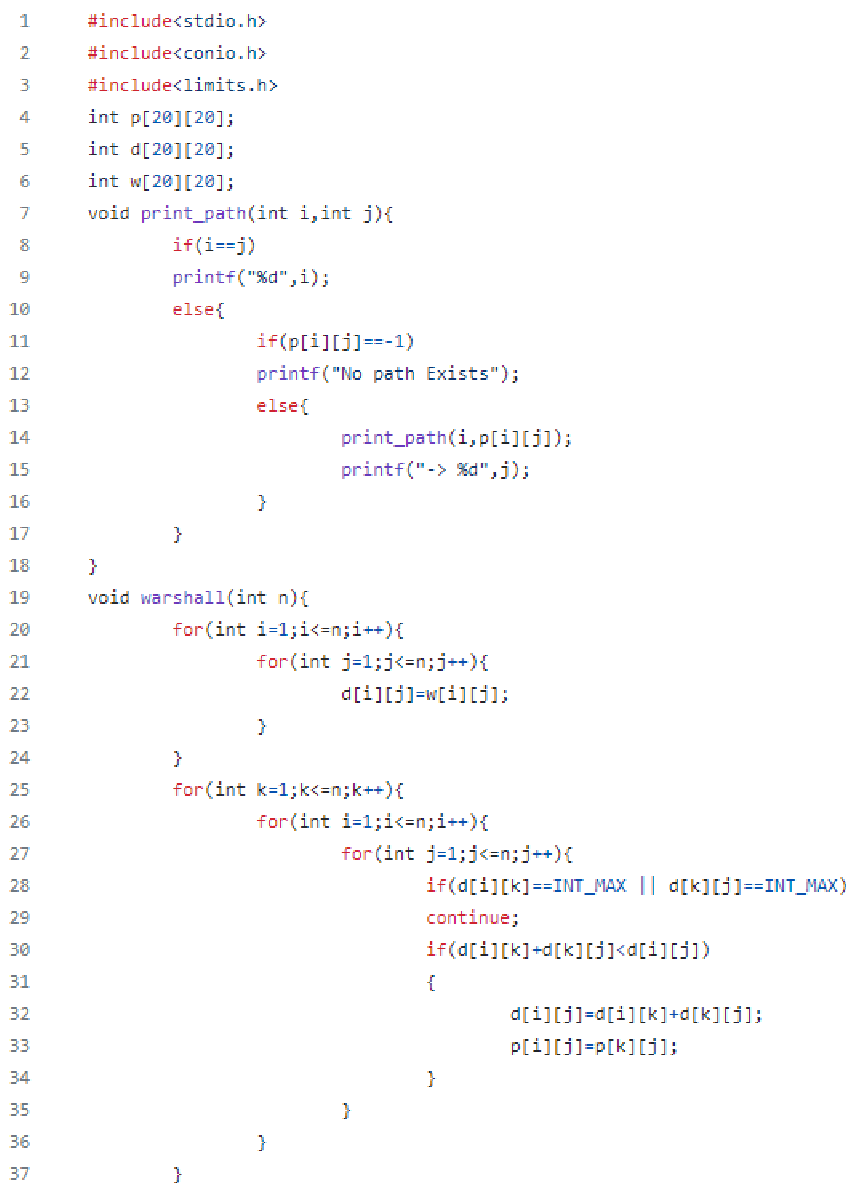


```c
#include<stdio.h>
#include<conio.h>
int temp;
void heap(int arr[10],int n,int i){
        int largest=i;
        int left=2*i+1;
        int right=2*i+2;
        if(left<n&& arr[left]>arr[largest])
        largest=left;
        if(right<n && arr[right]>arr[largest])
        largest=right;
        if(largest!=i){
                temp=arr[i];
                arr[i]=arr[largest];
                arr[largest]=temp;
                heap(arr,n,largest);
        }
}
void heapsort(int arr[],int  n){
        int i;
        for(i=n/2-1;i>=0;i--)
        heap(arr,n,i);
        for(i=n-1;i>=0;i--){
                temp=arr[0];
                arr[0]=arr[i];
                arr[i]=temp;
                heap(arr,i,0);
        }
}
void main(){
        int i,n,a[10];
        printf("enter the no. of element: ");
        scanf("%d",&n);
        printf("Enter elements: ");
        for(i=0;i<n;i++){
```



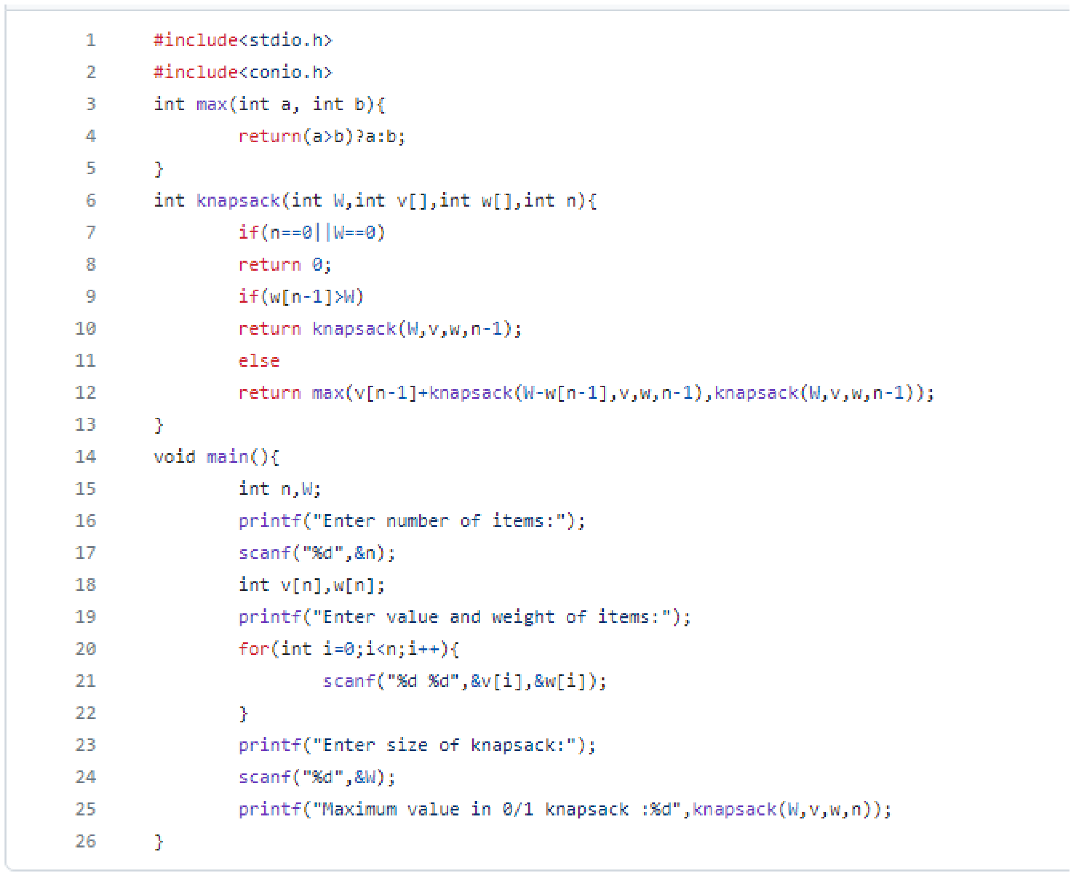
```c
                scanf("%d",&a[i]);
        }
        heapsort(a,n);
        for(i=0;i<n;i++){
                printf("%d\t",a[i]);
        }
}
```

FLOYD



```c
#include<stdio.h>
#include<conio.h>
#include<limits.h>
int p[20][20];
int d[20][20];
int w[20][20];
void print_path(int i,int j){
        if(i==j)
        printf("%d",i);
        else{
                if(p[i][j]==-1)
                printf("No path Exists");
                else{
                        print_path(i,p[i][j]);
                        printf("-> %d",j);
                }
        }
}
void warshall(int n){
        for(int i=1;i<=n;i++){
                for(int j=1;j<=n;j++){
                        d[i][j]=w[i][j];
                }
        }
        for(int k=1;k<=n;k++){
                for(int i=1;i<=n;i++){
                        for(int j=1;j<=n;j++){
                                if(d[i][k]==INT_MAX || d[k][j]==INT_MAX)
                                continue;
                                if(d[i][k]+d[k][j]<d[i][j])
                                {
                                        d[i][j]=d[i][k]+d[k][j];
                                        p[i][j]=p[k][j];
                                }
                        }
                }
        }
}
void main(){
        int i,j,v,s,des;
        char ch;
        printf("Enter number of vertices: ");
        scanf("%d",&v);
        printf("Enter the weight matrix");
        for(i=1;i<=v;i++)
        {
                for(j=1;j<=v;j++)
                {
                        if(i==j)
                        {
                                w[i][j]=0;
                                p[i][j]=-1;
                                continue;
                        }
                        printf("Is edge (%d,%d) present in graph (y/n): ",i,j);
                        fflush(stdin);
                        scanf("%c",&ch);
                        if(ch=='y' || ch=='Y'){
                                printf("Enter weight of edge (%d,%d): ",i,j);
                                scanf("%d",&w[i][j]);
                                p[i][j]=i;
                        }
                        else{
                                w[i][j]=INT_MAX;
                                p[i][j]=-1;
                        }
                }
        }
        warshall(v);
        printf("Enter source and destination: ");
        scanf("%d %d",&s,&des);
        printf("Distance = %d",d[s][des]);
        print_path(s,des);
}
```

KNAPSACK



```c
#include<stdio.h>
#include<conio.h>
int max(int a, int b){
        return(a>b)?a:b;
}
int knapsack(int W,int v[],int w[],int n){
        if(n==0||W==0)
        return 0;
        if(w[n-1]>W)
        return knapsack(W,v,w,n-1);
        else
        return max(v[n-1]+knapsack(W-w[n-1],v,w,n-1),knapsack(W,v,w,n-1));
}
void main(){
        int n,W;
        printf("Enter number of items:");
        scanf("%d",&n);
        int v[n],w[n];
        printf("Enter value and weight of items:");
        for(int i=0;i<n;i++){
                scanf("%d %d",&v[i],&w[i]);
        }
        printf("Enter size of knapsack:");
        scanf("%d",&W);
        printf("Maximum value in 0/1 knapsack :%d",knapsack(W,v,w,n));
}
```

## CHAIN MULTIPLICATION

```
#include<stdio.h>
#include<conio.h>
#include<limits.h>
int m[20][20],s[20][20];void Print_optimal_parens(i,j){
        if(i==j)
        {
                printf("A%d",i);

        }
        else
        {
                printf("(");
                Print_optimal_parens(i,s[i][j]);
                Print_optimal_parens(s[i][j]+1,j);
                printf(")");

        }

}
void Matrix_chain_order(int p[],int n){
        int q,j,i,l,k;
        for(i=1;i<=n;i++)
        {
                m[i][i]=0;

        }
        for(l=2;l<=n;l++)
        {
                for(i=1;i<=n-l+1;i++)
                {
```

```
            j=i+l-1;
            m[i][j]=INT_MAX;
            for(k=i;k<=j-1;k++)
            {
                q=m[i][k]+m[k+1][j]+p[i-1]*p[k]*p[j];
                if(q<m[i][j])
                {
                    m[i][j]=q;
                    s[i][j]=k;

                }

            }

        }

    }
    Print_optimal_parens(1,n);

}
void main(){
    int n;
    printf("enter the matrices");
    scanf("%d",&n);
    int p[n];
    for(int i=0;i<=n;i++)
    {
        scanf("%d",&p[i]);

    }
    Matrix_chain_order(p,n);
    printf("%d",m[1][n]);
}
```

## LARGEST SUBSEQUENCE

```c
#include<stdio.h>
#include<conio.h>
void lcs(char a[],char b[]){
        int n=strlen(a);
        int m=strlen(b);
        int c[n+1][m+1];
        for(int j=0;j<=m;j++){
                c[0][j]=0;
        }
        for(int i=1;i<=n;i++){
                c[i][0]=0;
        }
        for(int i=1;i<=n;i++){
                for(int j=1;j<=m;j++)
                {
                        if(a[i-1]==b[j-1])
                        c[i][j]=c[i-1][j-1]+1;
                        else if(c[i-1][j]>=c[i][j-1])
                        c[i][j]=c[i-1][j];
                        else
                        c[i][j]=c[i][j-1];
                }
        }
        printf("Length of LCS is %d\n",c[n][m]);
}
void main(){
        char a[50],b[50];
        printf("Enter a string1: ");
        gets(a);
        printf("Enter a string2: ");
        gets(b);
        lcs(a,b);
}
```

N QUEEN PROBLEM

```c
#include<stdio.h>
#include<conio.h>
int board[20],count;
int main(){
        int n,i,j;
        void queen(int row,int n);
        printf("Enter number of Queens: ");
        scanf("%d",&n);
        queen(1,n);
        return 0;
}

void print(int n){
        int i,j;
        for(i=1;i<=n;i++){
                for(j=1;j<=n;j++){
                        if(board[i]==j){
                                printf("row no %d\tcolom no %d\n",i,j);
                        }
                }
        }
}
int place(int row,int column){
        int i;
        for(i=1;i<=row-1;++i){
                if(board[i]==column){
                        return 0;
                }
                else if(abs(board[i]-column)==abs(i-row)){
                        return 0;
                }
        }
        return 1;
}
void queen(int row, int n){
        int column;
        for(column=1;column<=n;++column){
                if(place(row,column)){
                        board[row]=column;
                        if(row==n)
                        print(n);
                        else
                        queen(row+1,n);
                }
        }
}
```

PRIMS

```c
#include <stdio.h>
int a,b,u,v,n,i,j,ne=1;
int visited[10]= { 0 },min,mincost=0,cost[10][10];
int main(){
        printf("To compute the spanning tree from the adjacency matrix");
        printf("\nHow many nodes :");
        scanf("%d",&n);
        printf("Enter the adjacency matrix :");
        for (i=1;i<=n;i++)
        for (j=1;j<=n;j++){
                scanf("%d",&cost[i][j]);
                if(cost[i][j]==0)
                cost[i][j]=999;

        }
        printf("The entered adjacency matrix :\n");
        for(i=1;i<=n;i++){
                for(j=1;j<=n;j++){
                        if(cost[i][j]==999)
                        printf("%-3d",0);
                        else
                        printf("%-3d",cost[i][j]);
                }
                printf("\n");
        }
        visited[1]=1;
        printf("The nodes to be connected in spanning tree are : ");
        while(ne<n){
                for (i=1,min=999;i<=n;i++)
                for (j=1;j<=n;j++)
                if(cost[i][j]<min)
                if(visited[i]!=0)
                {
                        min=cost[i][j];
                        a=u=i;
                        b=v=j;

                }
                if(visited[u]==0 || visited[v]==0)
                {
                        printf("(%d,%d);",a,b);
                        ne++;
                        mincost+=min;
                        visited[b]=1;

                }
                cost[a][b]=cost[b][a]=999;

        }
        printf("\nThe cost of Minimum Spanning Tree is :%d",mincost);
        return 0;
}
```

KRUSKAL

```
#include<conio.h>
int parent[100];
int find(int i)
{
        while(parent[i]!=i)
        i=parent[i];
        return i;
}
void unio(int i,int j){
        int x,y;
        x=find(i);
        y=find(j);
        parent[x]=y;
}
void kruskal(int a[][100],int n){
        int k,co=0,min,r,b,l,res[100][2];
        for(k=0;k<n;k++)
        parent[k]=k;
        printf("The minimum spanning tree has the following edges:\n");
        while(co<n-1){
                min=10000000;
                r=-1;
                b=-1;
                for(k=n-1;k>-1;k--){
                        for(l=n-1;l>-1;l--){
                                if(find(k)!=find(l) && a[k][l]<min && a[k][l]!=0){
                                        min=a[k][l];
                                        r=k;
                                        b=l;
                                }
                        }
                }
                unio(r,b);
                res[co][0]=r+1;
                res[co][1]=b+1;
                co++;
        }
        for(k=n-2;k>-1;k--)
        printf("%d-%d\n",res[k][0],res[k][1]);
}
void main(){
        char c;
        int n,i,j,a[100][100],l[1000];
        printf("Input as adjacency matrix or adjacency list?(A/E)");
        scanf("%c",&c);
        printf("no of nodes :");
        scanf("%d",&n);
        printf("Input as adjacency matrix:\n");
        for(i=0;i<n;i++){
                printf("Row %d:",i+1);
                for(j=0;j<n;j++){
                        scanf("%d",&a[i][j]);
                }
        }
        kruskal(a,n);
}
```

KNAPSACK

```
#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=1;i<=n;i++){
                x[i]=0.0;
        }
        for(i=1;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=1;i<=n;i++)
        printf("%.2f\t",x[i]);
        printf("\nMaximum profit is:- %.2f",tp);
}
int main(){
        float weight[20],profit[20],capacity;
        int num,i,j;
        float ratio[20],temp;
        printf("Enter the no. of objects:- ");
        scanf("%d", &num);
        printf("Enter the Weight, Value(Profit) of each object:- \n");
        for(i=1;i<=num;i++){
                printf("item %d:",i);
                scanf("%f%f",&weight[i],&profit[i]);
        }
        printf("Enter the capacity of knapsack:- ");
        scanf("%f",&capacity);
        for(i=1;i<=num;i++){
                ratio[i]=profit[i]/weight[i];
        }
        for(i=1;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);
}
```