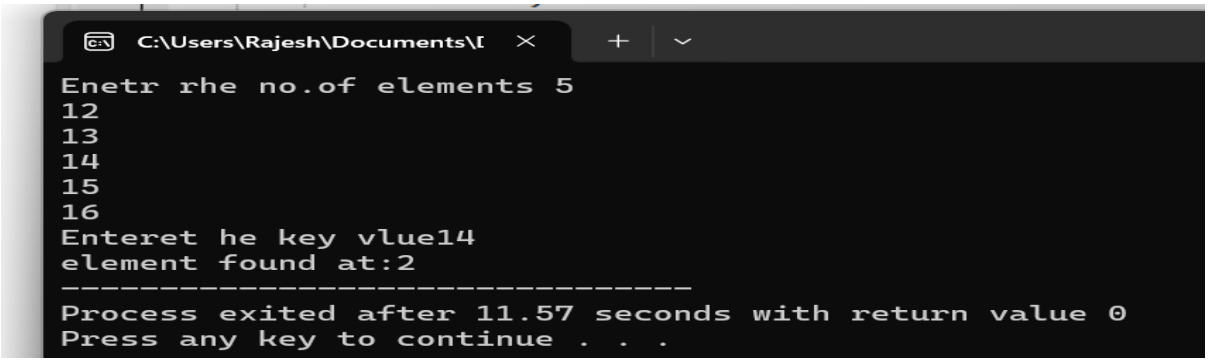
1) Binary search

```
#include<stdio.h>
int main()
{
        int a[10],key,flag;
        int n,i,low,high,mid;
        printf("Enetr rhe no.of elements ");
        scanf("%d",&n);
        printf("Enteret he key vlue");
        scanf("%d",&key);
        for(i=0;i<n;i++)
        {
                scanf("%d",&a[i]);
        }
        flag=0;
        low=0;
        high=n-1;
//      printf("%d",high);
        while(low<high)
        {
                mid=(low+high)/2;
        //      printf("%d",mid);
                if(a[mid]>key)
                {
                        high=mid-1;
                }
                else if(a[mid]<key)
                {
                        low=mid+1;
                }
                else if(a[mid]==key)
                {
                        flag=1;
                        break;
           }
   }
   if(flag==1)
   {
        printf("element found at:%d",mid);
        }
        else
        {
                printf("not found");
        }
   return 0;
}
```

Output:

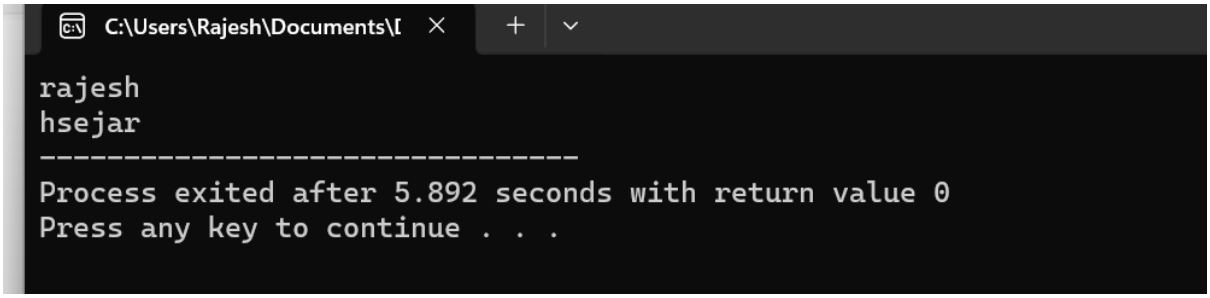


2) Reverse string using recursion

```
#include<stdio.h>
#include<string.h>
void rev(char s1[],char s[],int n,int i)
{
 static int j=0;
  if(i<=-1)
  {
      return;
  }
  else
  {
    s[j]=s1[i];
    j++;
      return rev(s1,s,n,i-1);
       }
}
int main()
{
      char s1[10],s[10];
      int n;
      scanf("%s",s1);
      n=strlen(s1);
```

```
        rev(s1,s,n,n-1);
        printf("%s",s);
    }
```

Output:



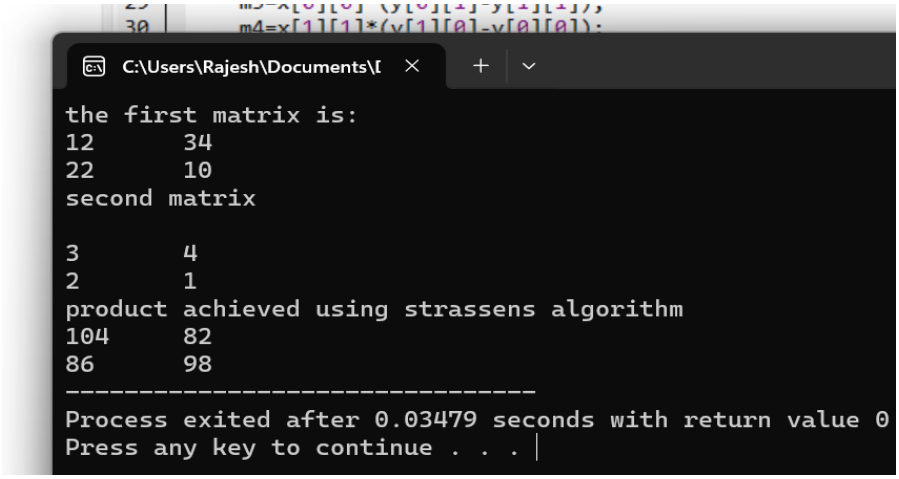
4)

Strassen multiplication:

```
#include<stdio.h>
int main()
{
        int z[2][2];
        int i,j;
        int m1,m2,m3,m4,m5,m6,m7;
        int x[2][2]={{12,34},{22,10}};
        int y[2][2]={{3,4},{2,1}};
        printf("the first matrix is:");
        for(i=0;i<2;i++)
        {
                printf("\n");
                for(j=0;j<2;j++)
                {
                        printf("%d\t",x[i][j]);
                }
        }
        printf("\nsecond matrix\n");
        for(i=0;i<2;i++)
        {
```

```
                printf("\n");
                for(j=0;j<2;j++)
                {
                        printf("%d\t",y[i][j]);
                }
        }
        m1=(x[0][0]+x[1][1])*(y[0][0]+y[1][1]);
        m2=(x[1][0]+x[1][1])*(y[0][0]);
        m3=x[0][0]*(y[0][1]-y[1][1]);
        m4=x[1][1]*(y[1][0]-y[0][0]);
        m5=(x[0][0]+x[0][1])*y[1][1];
        m6=(x[1][0]-x[0][0])*(y[0][0]+y[0][1]);
        m7=(x[0][1]-x[1][1])*(y[1][0]+y[1][1]);
        z[0][0]=m1+m4-m5+m7;
        z[0][1]=m3+m5;
        z[1][0]=m2+m4;
        z[1][1]=m1-m2+m3+m6;
        printf("\nproduct achieved using strassens algorithm");
        for(i=0;i<2;i++)
        {
                printf("\n");
                for(j=0;j<2;j++)
                {
                        printf("%d\t",z[i][j]);
                }
        }
        return 0;
}
```

Output:



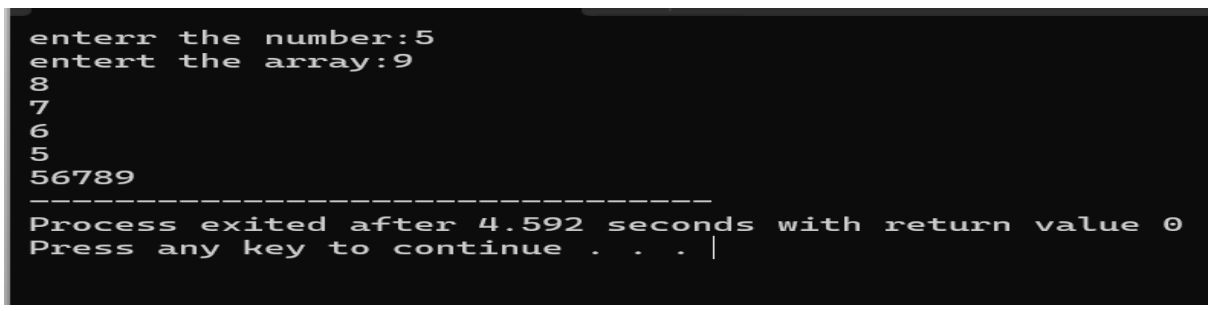
5)merge sort

```
#include<stdio.h>

void merge(int a[],int beg,int mid,int end)

{

        int i,j,k;

        int n1=mid-beg+1;

        int n2=end-mid;

        int left[n1],right[n2];

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

        {

                left[i]=a[beg+i];

  }

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

        {

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

  }

  i=0;

        j=0;

        k=beg;

        while(i<n1 && j<n2)

        {

                if(left[i]<=right[j])
```

```
                {
                        a[k]=left[i];
                        i++;
                }
                else
                {
                        a[k]=right[j];
                        j++;
                }
                k++;
        }
        while(i<n1)
        {
                a[k]=left[i];
                i++;
                k++;
        }
        while(j<n2)
        {
                a[k]=right[j];
                j++;
                k++;
        }
}
void display(int a[],int n)
{
        int i;
        for(i=0;i<n;i++)
        {
                printf("%d",a[i]);
        }
}
void mergesort(int a[],int beg,int end)
```

```
{
        if(beg<end)
        {
         int mid=(beg+end)/2;
        mergesort(a,beg,mid);
        mergesort(a,mid+1,end);
        merge(a,beg,mid,end);
  }
}
int main()
{
        int n,a[10],i,beg,end;
        printf("enterr the number:");
        scanf("%d",&n);
        printf("entert the array:");
        for(i=0;i<n;i++)
        {
                scanf("%d",&a[i]);
        }
        beg=0;
        end=n-1;
        mergesort(a,beg,end);
        display(a,n);
}
```

Output:

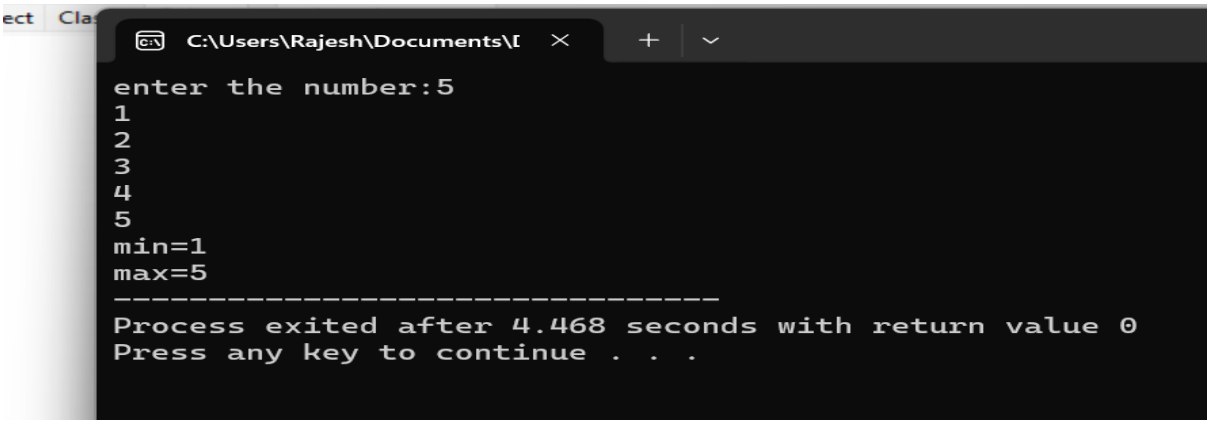


6) find min and max using divide and conquer

```
#include<stdio.h>
int min,max;
int a[100];
void maxmin(int i,int j)
{
	int max1,min1,mid;
	if(i==j)
	{
		max=min=a[i];
	}
	else if(i==j-1)
	{
		if(a[i]<a[j])
		{
			max=a[j];
			min=a[i];
		}
		else
		{
			max=a[i];
			min=a[j];
		}
	}
	else
	{
		mid=(i+j)/2;
		maxmin(i,mid);
		max1=max;
		min1=min;
		maxmin(mid+1,j);
		if(max<max1)
		{
```

```
                max=max1;
        }
        if(min>min1)
        {
                min=min1;
        }

    }
}
int main()
{
    int i,num;
    printf("enter the number:");
    scanf("%d",&num);
    for(i=1;i<=num;i++)
    {
        scanf("%d",&a[i]);
    }
    max=a[0];
    min=a[0];
    maxmin(1,num);
    printf("min=%d\n",min);
    printf("max=%d",max);
}
```

Output:

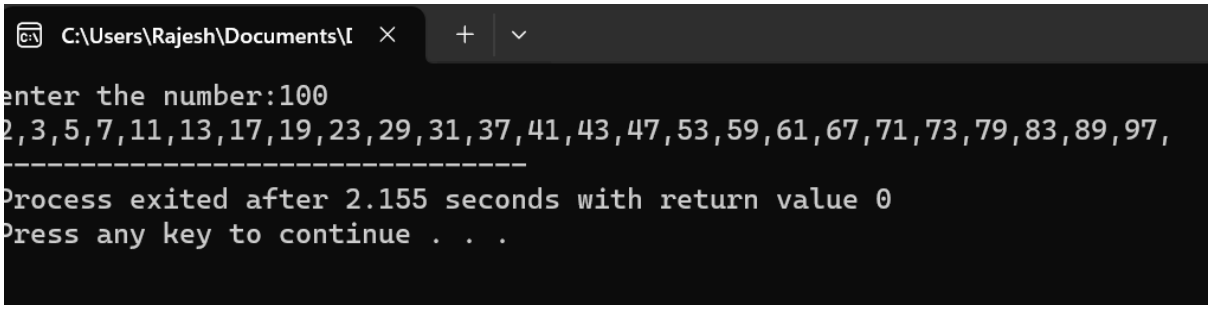


7)to generate all prime number using recursion

```
#include<stdio.h>
int checkprime(int i,int num)
{
	if(num==i)
	{
		return 0;
	}
	else
	{
		if(num%i==0)
		{
			return 1;
		}
	else
	{
		return checkprime(i+1,num);
	}
 }
}
int main()
{
	int n,i;
	printf("enter the number:");
	scanf("%d",&n);
	for(i=2;i<n;i++)
	{
		if(checkprime(2,i)==0)
		{
```

```
                printf("%d,",i);
            }
        }
}
```

Output:

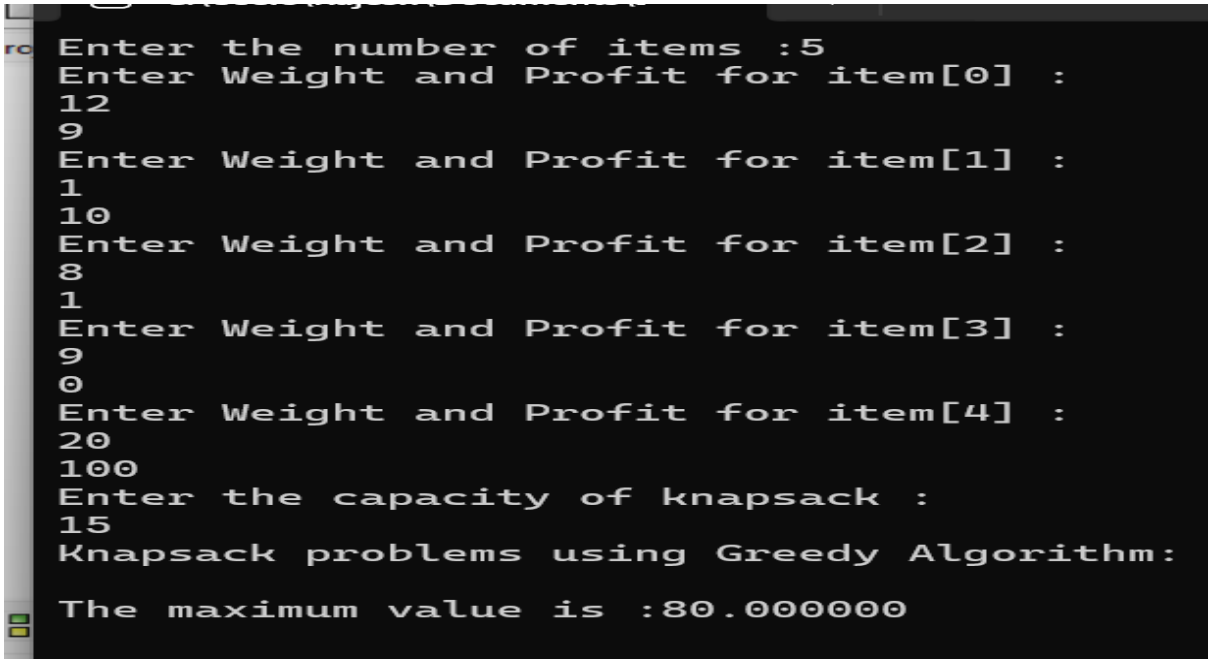


7) ksnapsack using greedy techniques

```
#include<stdio.h>
int main()
{
 float weight[50],profit[50],ratio[50],Totalvalue,temp,capacity,amount;
 int n,i,j;
 printf("Enter the number of items :");
 scanf("%d",&n);
 for (i = 0; i < n; i++)
 {
 printf("Enter Weight and Profit for item[%d] :\n",i);
 scanf("%f %f", &weight[i], &profit[i]);
 }
 printf("Enter the capacity of knapsack :\n");
 scanf("%f",&capacity);

 for(i=0;i<n;i++)
 ratio[i]=profit[i]/weight[i];

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

printf("Knapsack problems using Greedy Algorithm:\n");
for (i = 0; i < n; i++)
{
if (weight[i] > capacity)
break;
else
{
Totalvalue = Totalvalue + profit[i];
capacity = capacity - weight[i];
}
}
if (i < n)
Totalvalue = Totalvalue + (ratio[i]*capacity);
printf("\nThe maximum value is :%f\n",Totalvalue);
return 0;
}
```

Output:

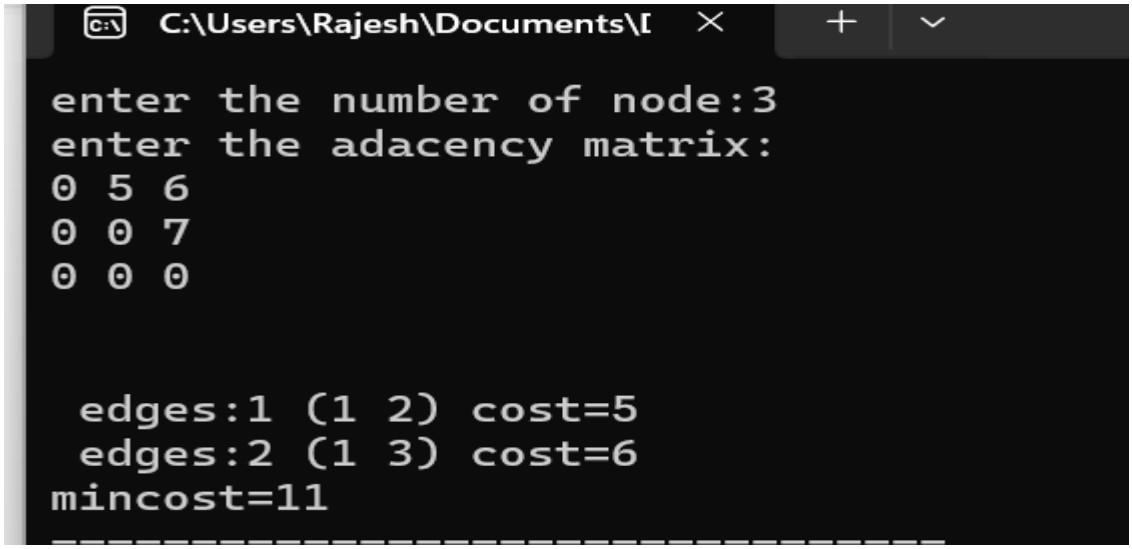


8)MST using Kruskal algorithm

```
#include<stdio.h>
int a,b,j,i,n,u,v,ne=1;
int min,cost[10][10];
int visited[10]={0},mincost=0;
int main()
{
	printf("enter the number of node:");
	scanf("%d",&n);
	printf("enter the adacency 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;
			}
```

```
        }
        }
        visited[1]=1;
        printf("\n");
        while(ne<n)
        {
                min=999;
        for(i=1;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("\n edges:%d (%d %d) cost=%d",ne++,a,b,min);
                                        mincost+=min;
                                        visited[b]=1;
                                }
                                cost[a][b]=cost[b][a]=999;
                        }
                }
        }
}
printf("\nmincost=%d",mincost);}
```

output


C:\Users\Rajesh\Documents\[
enter the number of node:3
enter the adacency matrix:
0 5 6
0 0 7
0 0 0
edges:1 (1 2) cost=5
edges:2 (1 3) cost=6
mincost=11