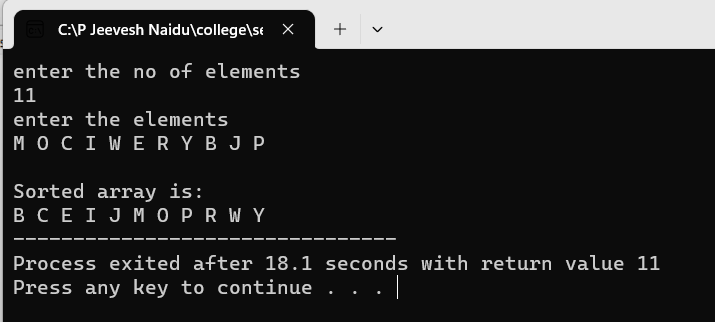
Program 1:

#include <stdio.h>  
  
char a[50];  
  
void merge(int low,int mid,int high){  
 char b[50];  
 int i=low,j=mid+1,k=low;  
 while(i<=mid && j<=high){  
 if(a[i]<a[j])  
 b[k++]=a[i++];  
 else  
 b[k++]=a[j++];  
 }  
 while(i<=mid)  
 b[k++]=a[i++];  
 while(j<=high)  
 b[k++]=a[j++];  
 for(i=low;i<=high;i++)  
 a[i]=b[i];  
}  
  
void mergesort(int low,int high){  
 int mid;  
 if(low<high){  
 mid=(low+high)/2;  
 mergesort(low,mid);  
 mergesort(mid+1,high);  
 merge(low,mid,high);  
 }  
}  
  
int binarysearch(int low, int high, char c) {  
 if(low==high) {  
 if(a[low]==c)  
 return low;  
 else   
 return 0;  
 }  
 int mid=(low+high)/2;  
 if(a[mid]==c)  
 return mid;  
 else if (c>a[mid])  
 return binarysearch(mid,high,c);  
 else  
 return binarysearch(low,mid-1,c);   
}  
  
int main(){  
   
 int n,s,p,i;  
 printf("enter the no of elements\n");  
 scanf("%d",&n);  
 printf("enter the elements\n");  
 for(i=0;i<n;i++){  
 scanf(" %c",&a[i]);  
 }  
 mergesort(0,n-1);  
 printf("\nSorted array is:\n");  
 for(i=0;i<n;i++){  
 printf("%c ",a[i]);  
 }  
}

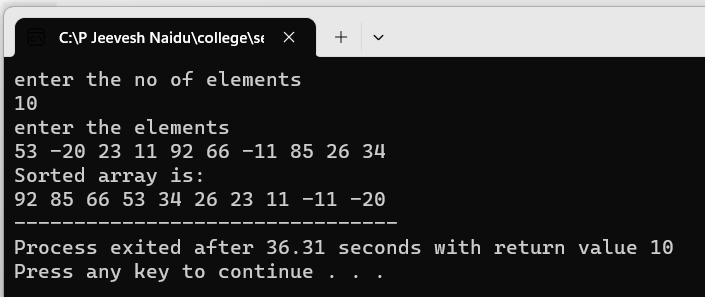
Output:



Program 2:

#include <stdio.h>  
  
int a[50];  
  
void merge(int low,int mid,int high){  
 int b[50];  
 int i=low,j=mid+1,k=low;  
 while(i<=mid && j<=high){  
 if(a[i]>a[j])  
 b[k++]=a[i++];  
 else  
 b[k++]=a[j++];  
 }  
 while(i<=mid)  
 b[k++]=a[i++];  
 while(j<=high)  
 b[k++]=a[j++];  
 for(i=low;i<=high;i++)  
 a[i]=b[i];  
}  
  
void mergesort(int low,int high){  
 int mid;  
 if(low<high){  
 mid=(low+high)/2;  
 mergesort(low,mid);  
 mergesort(mid+1,high);  
 merge(low,mid,high);  
 }  
}  
  
int main(){  
   
 int n,s,p,i;  
 printf("enter the no of elements\n");  
 scanf("%d",&n);  
 printf("enter the elements\n");  
 for(i=0;i<n;i++){  
 scanf("%d",&a[i]);  
 }  
 mergesort(0,n-1);  
 printf("Sorted array is:\n");  
 for(i=0;i<n;i++){  
 printf("%d ",a[i]);  
 }  
}

Output:



Program 3:

#include <stdio.h>

#include <stdlib.h>

#define sint(x) scanf("%d", &x);

#define ch(x) scanf(" %c", &x);

void charmerge(char arr[], int left, int mid, int right)

{

int i, j, k;

int n1 = mid - left + 1;

int n2 = right - mid;

char L[n1], R[n2];

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

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

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

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

i = 0;

j = 0;

k = left;

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 charmergesort(char arr[], int left, int right)

{

if (left < right)

{

int mid = left + (right - left) / 2;

charmergesort(arr, left, mid);

charmergesort(arr, mid + 1, right);

charmerge(arr, left, mid, right);

}

}

int bs(char arr[], char item, int low, int up)

{

int mid;

if (low > up)

return -1;

mid = (low + up) / 2;

if (item < arr[mid])

bs(arr, item, low, mid - 1);

else if (item > arr[mid])

bs(arr, item, mid + 1, up);

else

return mid;

}

int main()

{

char ch;

int charsize,i;

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

sint(charsize);

char charr[charsize];

printf("Enter the character array\n");

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

ch(charr[i]);

charmergesort(charr, 0, charsize - 1);

while (1)

{

int c;

printf("Enter 1 to search\n2 to exit\n");

sint(c);

if (c == 2)

exit(1);

else

{

printf("Enter the character to be searched: ");

ch(ch);

if (bs(charr, ch, 0, charsize - 1) == -1)

printf("%c not found\n", ch);

else

printf("%c found\n", ch);

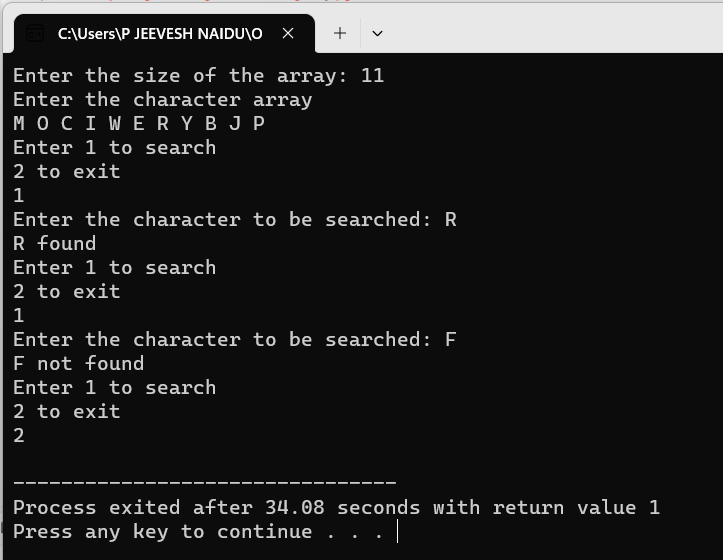
}

}

return 0;

}

Output:



Program 4:

#include<stdio.h>

#define MAX 100

int a[MAX];

void minmax(int i,int j,int \*max,int \*min)

{

if(i==j)

\*max=\*min=a[i];

else if(i==j-1)

{

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

{

\*max=a[i];

\*min=a[j];

}

else

{

\*max=a[j];

\*min=a[i];

}

}

else

{

int mid=(i+j)/2;

int max1,min1;

minmax(i,mid,max,min);

minmax(mid+1,j,&max1,&min1);

if(max1>\*max)

\*max=max1;

if(min1<\*min)

\*min=min1;

}

}

int main()

{

int min,max,n,i;

printf("enter the no of elements - ");

scanf("%d",&n);

printf("enter the elements\n");

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

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

}

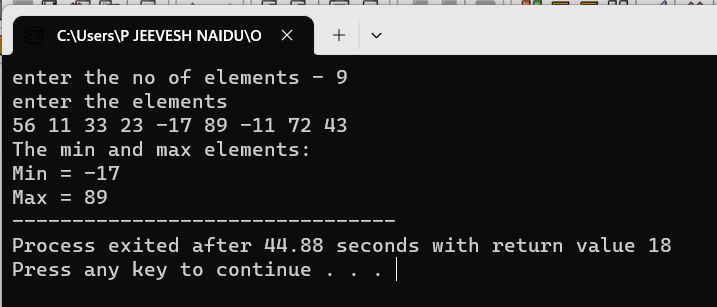
printf("The min and max elements:\n");

minmax(0,n-1,&max,&min);

printf("Min = %d\nMax = %d",min,max);

}

Output:



Program 5:

#include <stdio.h>

#define sint(x) scanf("%d", &x);

// utility functions for qs

int size, count = 1;

void sw(int a[], int i, int j)

{

int t = a[i];

a[i] = a[j], a[j] = t;

}

void dis(int a[], int s)

{

int i;

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

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

printf("\n");

}

int part(int a[], int l, int r)

{

int j;

int piv = a[r], i = l; // we choose the rightmost element is the pivot

for (j = l; j < r; j++)

{

if (a[j] < piv)

{

sw(a, i, j);

i++;

}

}

sw(a, i, r);

return i;

}

void qs(int a[], int l, int r)

{

// step 1: find the partition index

if (l < r)

{

int p = part(a, l, r);

printf("pass %d: ", count), count++;

dis(a, size);

qs(a, l, p - 1);

qs(a, p + 1, r);

}

}

int main()

{

int i;

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

sint(size);

int a[size];

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

sint(a[i]);

printf("Original Array: ");

dis(a, size);

qs(a, 0, size - 1);

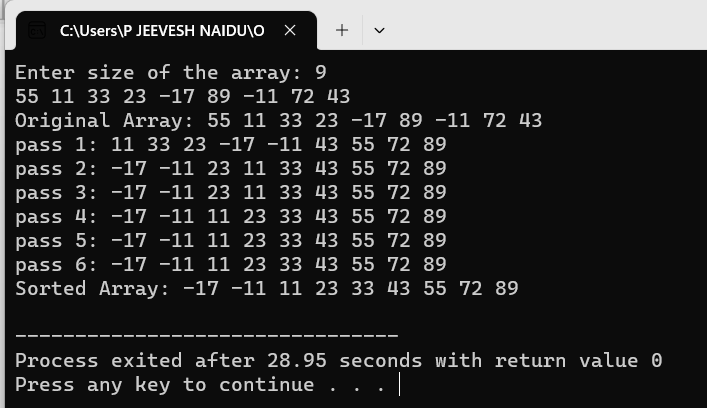
printf("Sorted Array: ");

dis(a, size);

return 0;

}

Output:



Program 6:

#include <stdio.h>

#define sint(x) scanf("%d", &x);

int size;

// utility functions

void sw(int a[], int i, int j)

{

int t = a[i];

a[i] = a[j], a[j] = t;

}

void dis(int a[], int s)

{

int i;

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

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

printf("\n");

}

int part(int a[], int l, int r)

{

int j;

int piv = a[r], i = l; // we choose the rightmost element is the pivot

for (j = l; j < r; j++)

{

if (a[j] < piv)

{

sw(a, i, j);

i++;

}

}

sw(a, i, r);

dis(a, size);

return i;

}

void ksmall(int a[], int l, int r, int x)

{

if (r < x)

{

printf("out of bounds\n");

return;

}

do

{

int p = part(a, l, r);

printf("pivot: %d\n", p);

if (p == x)

{

printf("The kth smallest element is: %d\n", a[p]);

return;

}

else if (p > x)

r = p - 1;

else

l = p + 1;

} while (l <= r);

printf("The kth smallest element is not present\n");

return;

// if (l <= r)

// {

// int p = part(a, l, r);

// printf("pivot: %d\n", p);

// if (p == x)

// {

// printf("%dth smallest element is: %d\n", ++x, a[p]);

// return;

// }

// else if (p > x)

// ksmall(a, l, p - 1, x);

// else if (p < x)

// ksmall(a, p + 1, r, x);

// }

// else

// printf("The element is not present\n");

}

int main()

{

int x,i;

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

sint(size);

int a[size];

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

sint(a[i]);

// int a[] = {42, 5, 90, 33, 28, 88, 56, 54, 11, 23, 76};

// size = sizeof(a) / sizeof(a[0]);

while (x != -1)

{

printf("Enter the kth value to be searched: ");

sint(x);

x--;

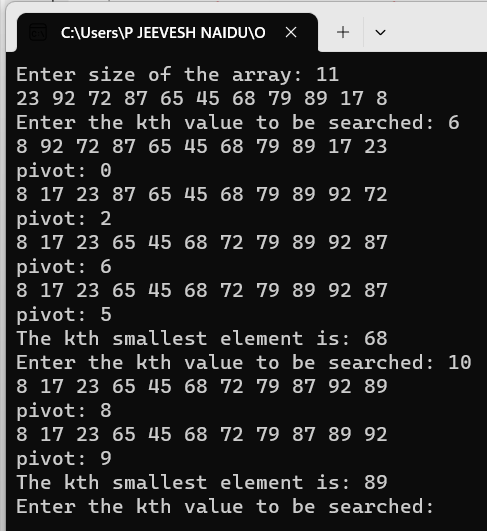
ksmall(a, 0, size - 1, x);

}

return 0;

}

Output:



Program 7:

#include <stdio.h>

#define sch(x) scanf(" %c", &x);

#define sint(x) scanf("%d", &x);

int size;

void sw(char a[], int i, int j)

{

char t = a[i];

a[i] = a[j], a[j] = t;

}

void dis(char a[], int s)

{

int i;

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

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

printf("\n");

}

int part(char a[], int l, int r)

{

int j;

int piv = a[r], i = l; // we choose the rightmost element is the pivot

for (j = l; j < r; j++)

{

if (a[j] < piv)

{

sw(a, i, j);

i++;

}

}

sw(a, i, r);

dis(a, size);

return i;

}

void ksmall(char a[], int l, int r, int x)

{

if (r < x)

{

printf("out of bounds\n");

return;

}

do

{

int p = part(a, l, r);

printf("pivot: %d\n", p);

if (p == x)

{

printf("The kth smallest element is: %c\n", a[p]);

return;

}

else if (p > x)

r = p - 1;

else

l = p + 1;

} while (l <= r);

printf("The kth smallest element is not present\n");

return;

// if (l <= r)

// {

// int p = part(a, l, r);

// printf("pivot: %d\n", p);

// if (p == x)

// {

// printf("%dth smallest element is: %c\n", ++x, a[p]);

// return;

// }

// else if (p > x)

// ksmall(a, l, p - 1, x);

// else if (p < x)

// ksmall(a, p + 1, r, x);

// }

// else

// printf("The element is not present\n");

}

int main()

{

// int size;

int i;

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

sint(size);

char a[size];

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

sch(a[i]);

// int a[]={42, 5, 90, 33, 28, 88, 56, 54, 11, 23, 76};

// size=sizeof(a)/sizeof(a[0]);

int x;

while (x != -1)

{

printf("\nEnter the kth value to be searched: ");

sint(x);

x--;

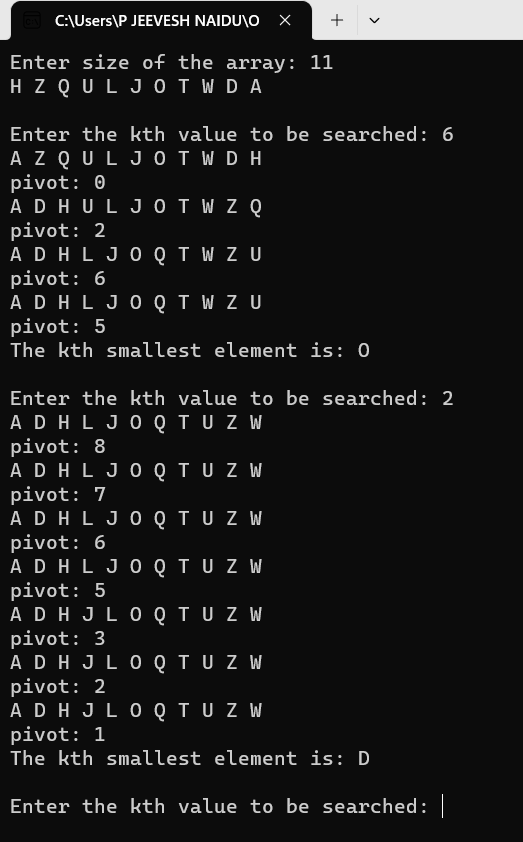
ksmall(a, 0, size - 1, x);

}

return 0;

}

Output:



Program 8:

#include <stdio.h>

#define sint(x) scanf("%d", &x);

#define new printf("\n");

#define N 8

void sa(int a[N][N], int b[N][N], int c[N][N], int n, int x)

{

int i,j;

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

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

c[i][j] = x ? a[i][j] - b[i][j] : a[i][j] + b[i][j];

}

void copy(int a[N][N], int b[N][N], int r1, int r2, int c1, int c2)

{

int i,j,x,y;

int size = r2 - r1;

for (i = r1, j = 0; i < r2 && j < size; i++, j++)

for (x = c1, y = 0; x < c2 && y < size; x++, y++)

b[j][y] = a[i][x];

}

void combine(int C11[N][N], int C12[N][N], int C21[N][N], int C22[N][N], int C[N][N], int n)

{

int i,j,k,l;

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

for (j = 0; j < n / 2; j++)

C[i][j] = C11[i][j];

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

for (j = 0, k = n / 2; j < n / 2 && k < n; j++, k++)

C[i][k] = C12[i][j];

for (i = 0, k = n / 2; i < n / 2 && k < n; i++, k++)

for (j = 0; j < n / 2; j++)

C[k][j] = C21[i][j];

for (i = 0, k = n / 2; i < n / 2 && k < n; i++, k++)

for (j = 0, l = n / 2; j < n / 2 && l < n; j++, l++)

C[k][l] = C22[i][j];

}

void strassen(int a[N][N], int b[N][N], int c[N][N], int n)

{

// printf("entered strassen\n");

if (n == 2)

{

// printf("entered base case\n");

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

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

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

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

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

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

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

c[0][0] = P + S - T + V;

c[0][1] = R + T;

c[1][0] = Q + S;

c[1][1] = P + R - Q + U;

}

else

{

// here we first find mid

// then we make 4 new matrices each for a, b and c

// we copy parts from the original matrix to these new matrices

// we find p, q ...but call strassen fn on parts of these

// finally combine into orig c

int mid = n / 2;

int a11[N][N], a12[N][N], a21[N][N], a22[N][N], b11[N][N], b12[N][N], b21[N][N], b22[N][N], c11[N][N], c12[N][N], c21[N][N], c22[N][N];

int p[N][N], q[N][N], r[N][N], s[N][N], t[N][N], u[N][N], v[N][N];

int t1[N][N], t2[N][N]; // temp matrices will be used to store intermediate results between recusive calls

// copying part

// here to copy for 11 and 12 the rows remain constant and the colms change

copy(a, a11, 0, mid, 0, mid); // here the rows are 0 to mid because they are const and cols change

copy(a, a12, 0, mid, mid, n);

copy(a, a21, mid, n, 0, mid);

copy(a, a22, mid, n, mid, n);

// same for b

copy(b, b11, 0, mid, 0, mid);

copy(b, b12, 0, mid, mid, n);

copy(b, b21, mid, n, 0, mid);

copy(b, b22, mid, n, mid, n);

sa(a11, a22, t1, mid, 0), sa(b11, b22, t2, mid, 0), strassen(t1, t2, p, mid); // P

sa(a21, a22, t1, mid, 0), strassen(t1, b11, q, mid); // Q

sa(b12, b22, t2, mid, 1), strassen(a11, t2, r, mid); // R

sa(b21, b11, t2, mid, 1), strassen(a22, t2, s, mid); // S

sa(a11, a12, t1, mid, 0), strassen(t1, b22, t, mid); // T

sa(a21, a11, t1, mid, 1), sa(b11, b12, t2, mid, 0), strassen(t1, t2, u, mid); // U

sa(a12, a22, t1, mid, 1), sa(b21, b22, t2, mid, 0), strassen(t1, t2, v, mid); // V

// add to c11, c12 ...

sa(p, s, t1, mid, 0), sa(v, t, t2, mid, 1), sa(t1, t2, c11, mid, 0); // c11

sa(r, t, c12, mid, 0); // c12

sa(q, s, c21, mid, 0); // c21

sa(p, r, t1, mid, 0), sa(u, q, t2, mid, 1), sa(t1, t2, c22, mid, 0); // c22

// now we combine the parts

combine(c11, c12, c21, c22, c, n);

}

}

int main()

{

int n = 1,i,j;

while (n % 2 != 0)

{

printf("Enter the size (order 2x): ");

sint(n);

}

int a[N][N], b[N][N], c[N][N];

printf("Enter the elements of matrix a\n");

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

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

sint(a[i][j]);

printf("Enter the elements of matrix b\n");

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

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

sint(b[i][j]);

strassen(a, b, c, n);

printf("The product of the two matrices is\n");

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

{

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

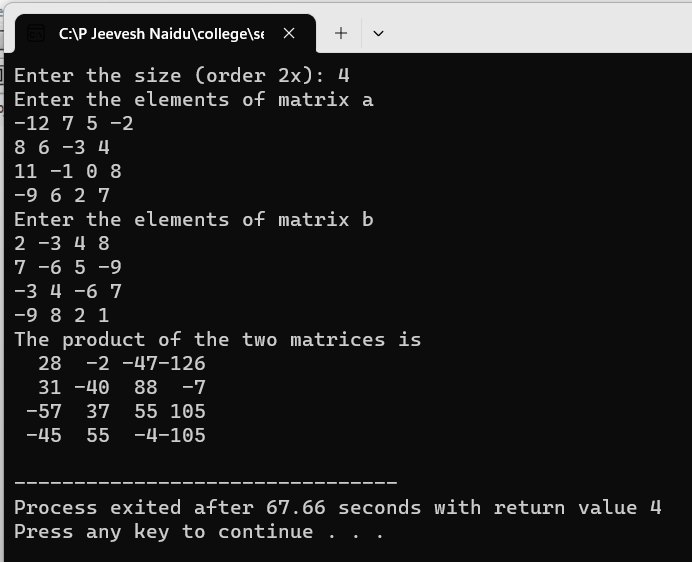
printf("%4d", c[i][j]);

new;

}

}

Output:



Conclusion: All programs were successfully executed.