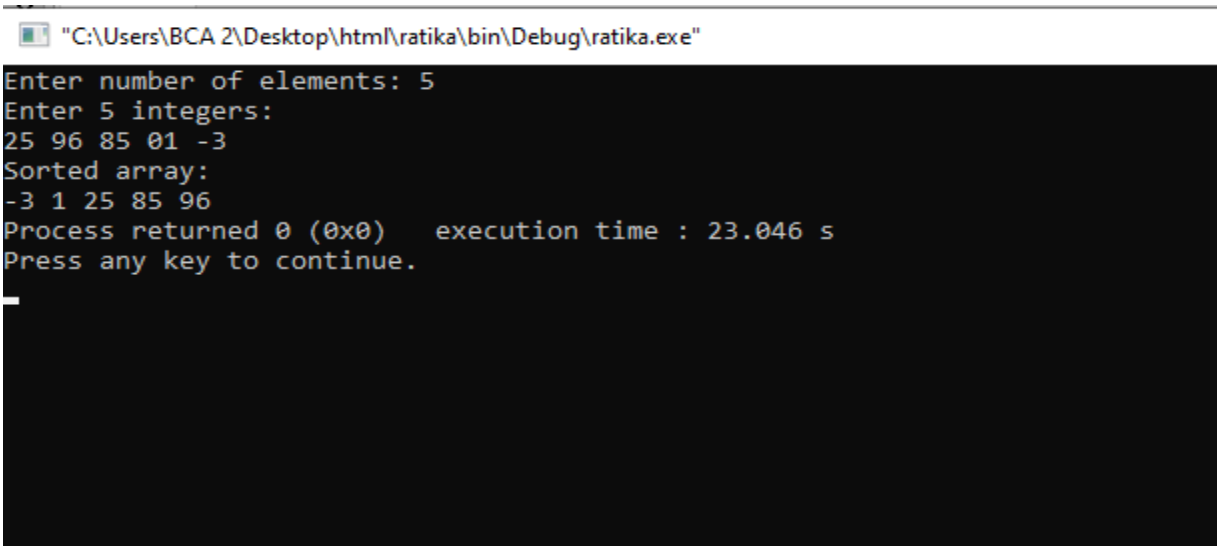
DAA LAB

1. Write a program to sort a list of N elements using Selection Sort Technique.

```
#include <stdio.h>
void selectionSort(int a[], int n) {
    for (int i = 0; i < n - 1; i++) {
        int min = i;
        for (int j = i + 1; j < n; j++)
            if (a[j] < a[min]) min = j;
        int temp = a[i];
      a[i] = a[min];
a[min] = temp;
    }
}

int main() {
 int a[100], n;
 printf("Enter number of elements: ");
 scanf("%d", &n);
 printf("Enter %d integers:\n", n);
 for (int i = 0; i < n; i++)
     scanf("%d", &a[i]);
 selectionSort(a, n);
 printf("Sorted array:\n");
 for (int i = 0; i < n; i++)
printf("%d ", a[i]);
    return 0;
}
```

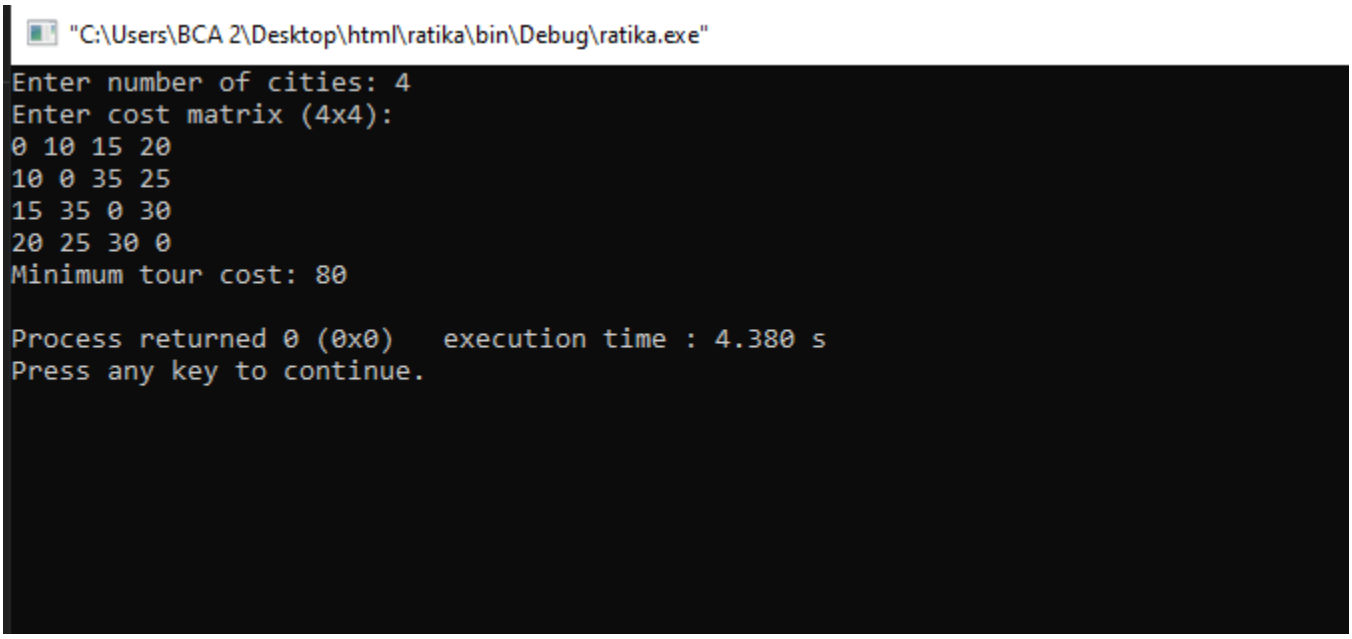
OUTPUT :



2. Write a program to perform Travelling Salesman .

```
#include <stdio.h>
#include <limits.h>
#define MAX 10

int tsp(int graph[MAX][MAX], int visited[MAX], int pos, int n, int count, int cost, int start) {
   if (count == n && graph[pos][start])
      return cost + graph[pos][start];

   int ans = INT_MAX;
   for (int i = 0; i < n; i++) {
      if (!visited[i] && graph[pos][i]) {
         visited[i] = 1;
         int temp = tsp(graph, visited, i, n, count + 1, cost + graph[pos][i], start);
         if (temp < ans) ans = temp;
         visited[i] = 0;
      }
   }
   return ans;
}

int main() {
   int graph[MAX][MAX], visited[MAX] = {0}, n;
   printf("Enter number of cities: ");
   scanf("%d", &n);
   printf("Enter cost matrix (%dx%d):\n", n, n);
   for (int i = 0; i < n; i++)
      for (int j = 0; j < n; j++)
         scanf("%d", &graph[i][j]);

   visited[0] = 1;
   int minCost = tsp(graph, visited, 0, n, 1, 0, 0);
   printf("Minimum tour cost: %d\n", minCost);
   return 0;
}
```

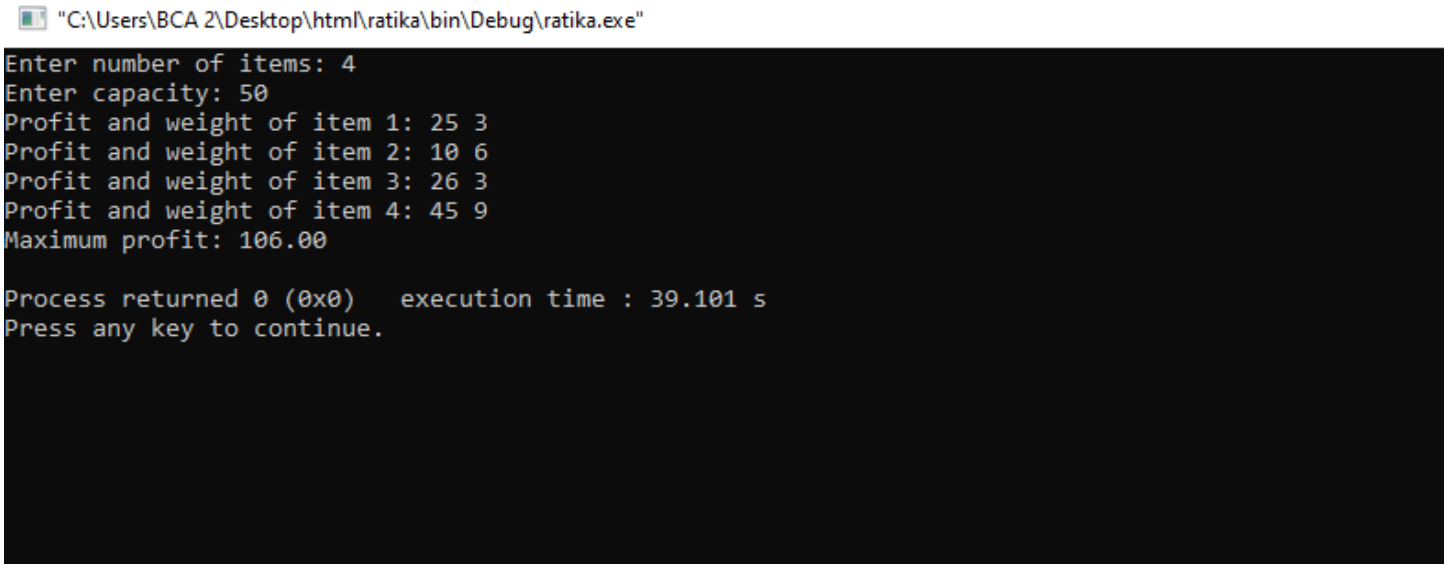
OUTPUT :


"C:\Users\BCA 2\Desktop\html\ratika\bin\Debug\ratika.exe"
Enter number of cities: 4
Enter cost matrix (4x4):
0 10 15 20
10 0 35 25
15 35 0 30
20 25 30 0
Minimum tour cost: 80
Process returned 0 (0x0)   execution time : 4.380 s
Press any key to continue.

3. Write a program to perform Knapsack Problem using Greedy Solution?

```
#include <stdio.h>

typedef struct { float w, p, r; } Item;

void sort(Item a[], int n) {
    for (int i = 0; i < n-1; i++)
        for (int j = i+1; j < n; j++)
            if (a[i].r < a[j].r) {
                Item t = a[i]; a[i] = a[j]; a[j] = t;
            }
}

int main() {
    int n;
   float cap, total = 0;
    Item a[100];
    printf("Enter number of items: "); scanf("%d", &n);
    printf("Enter capacity: "); scanf("%f", &cap);
    for (int i = 0; i < n; i++) {
        printf("Profit and weight of item %d: ", i+1);
        scanf("%f%f", &a[i].p, &a[i].w);
        a[i].r = a[i].p / a[i].w;
    }

    sort(a, n);
    for (int i = 0; i < n && cap > 0; i++) {
        if (a[i].w <= cap) {
            total += a[i].p;
            cap -= a[i].w;
        } else {
            total += a[i].r * cap;
            break;
        }
    }
    printf("Maximum profit: %.2f\n", total);
    return 0;
}
```

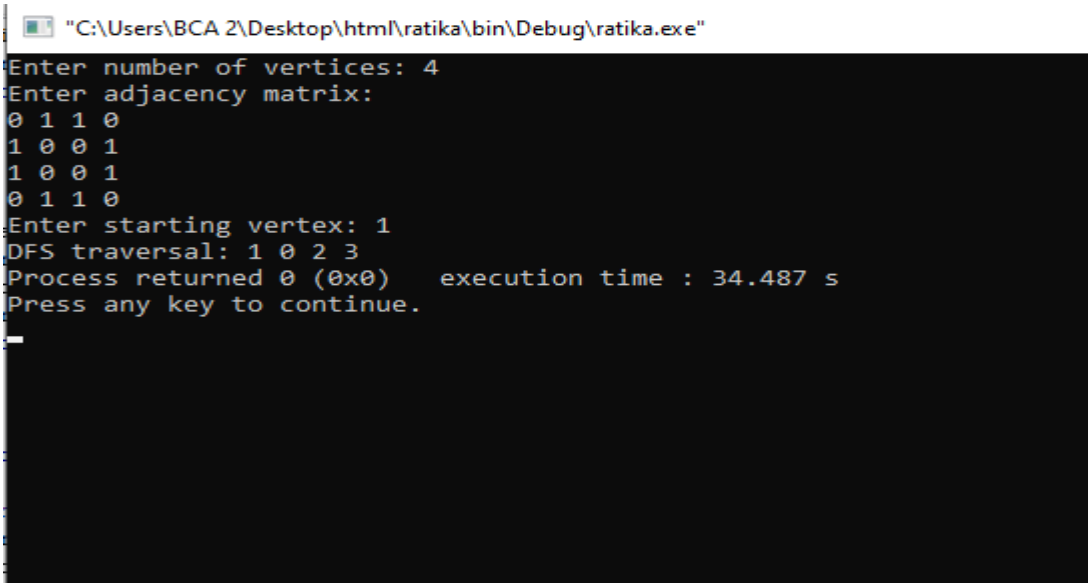
OUTPUT:


"C:\Users\BCA 2\Desktop\html\ratika\bin\Debug\ratika.exe"
Enter number of items: 4
Enter capacity: 50
Profit and weight of item 1: 25 3
Profit and weight of item 2: 10 6
Profit and weight of item 3: 26 3
Profit and weight of item 4: 45 9
Maximum profit: 106.00
Process returned 0 (0x0)   execution time : 39.101 s
Press any key to continue.

4. Write program to implement the DFS algorithm for a graph.

```
#include <stdio.h>
int graph[10][10], visited[10], n;
void DFS(int v) {
   visited[v] = 1;
   printf("%d ", v);
   for (int i = 0; i < n; i++)
      if (graph[v][i] && !visited[i])
         DFS(i);
}
int main() {
   printf("Enter number of vertices: ");
   scanf("%d", &n);
   printf("Enter adjacency matrix:\n");
   for (int i = 0; i < n; i++)
      for (int j = 0; j < n; j++)
         scanf("%d", &graph[i][j]);

   for (int i = 0; i < n; i++) visited[i] = 0;

   int start;
   printf("Enter starting vertex: ");
   scanf("%d", &start);
   printf("DFS traversal: ");
   DFS(start);
   return 0;
}
```

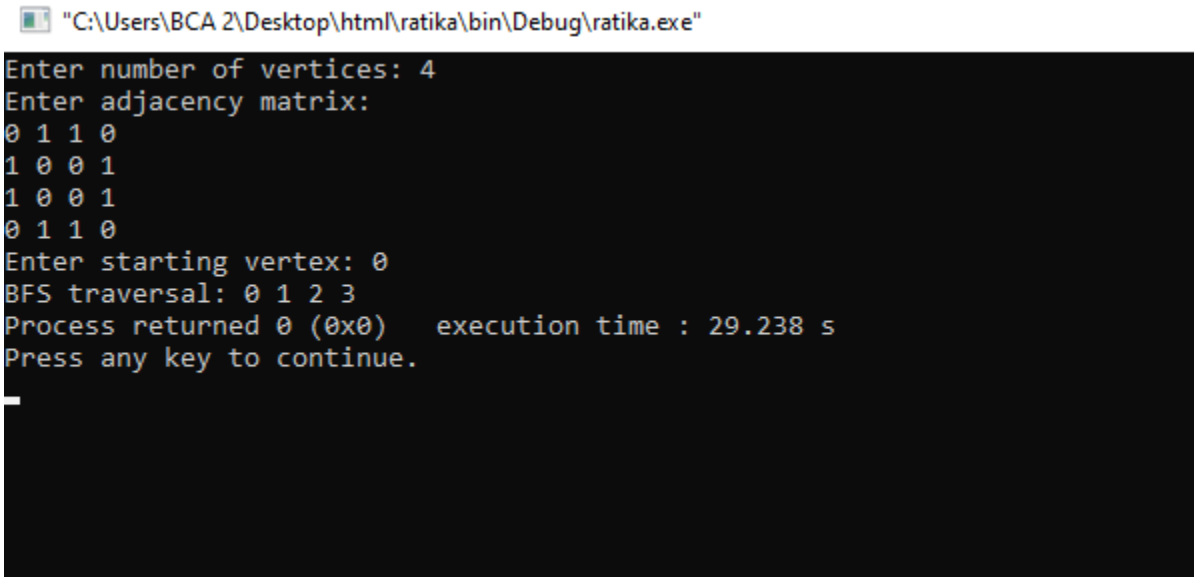
OUTPUT:



5. Write program to implement the BFS algorithm for a graph.

```
#include <stdio.h>
int graph[10][10], visited[10], queue[10], n, front = 0, rear = -1;

void BFS(int start) {
   visited[start] = 1;
   queue[++rear] = start;
   while (front <= rear) {
      int v = queue[front++];
      printf("%d ", v);
      for (int i = 0; i < n; i++)
         if (graph[v][i] && !visited[i]) {
            visited[i] = 1;
            queue[++rear] = i;
         }
   }
}

int main() {
   printf("Enter number of vertices: ");
   scanf("%d", &n);
   printf("Enter adjacency matrix:\n");
   for (int i = 0; i < n; i++)
      for (int j = 0; j < n; j++)
         scanf("%d", &graph[i][j]);
   for (int i = 0; i < n; i++) visited[i] = 0;

   int start;
   printf("Enter starting vertex: ");
   scanf("%d", &start);
   printf("BFS traversal: ");
   BFS(start);
   return 0;
}
```

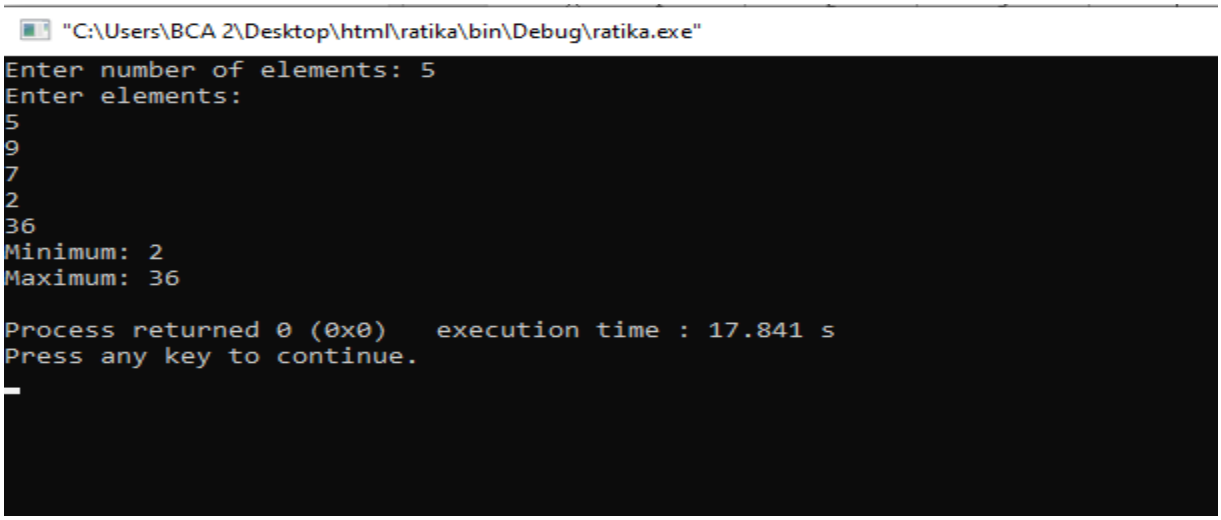
OUTPUT :


"C:\Users\BCA 2\Desktop\html\ratika\bin\Debug\ratika.exe"
Enter number of vertices: 4
Enter adjacency matrix:
0 1 1 0
1 0 0 1
1 0 0 1
0 1 1 0
Enter starting vertex: 0
BFS traversal: 0 1 2 3
Process returned 0 (0x0)   execution time : 29.238 s
Press any key to continue.

6. Write a program to find minimum and maximum value in an array using divide and conquer.

```
#include <stdio.h>
void findMinMax(int a[], int low, int high, int *min, int *max) {
   if (low == high)
      *min = *max = a[low];
   else if (high == low + 1) {
      if (a[low] < a[high]) {*min = a[low]; *max = a[high];}
      else {*min = a[high]; *max = a[low];}
   } else {
      int mid = (low + high) / 2, min1, max1, min2, max2;
      findMinMax(a, low, mid, &min1, &max1);
      findMinMax(a, mid+1, high, &min2, &max2);
      *min = (min1 < min2) ? min1 : min2;
      *max = (max1 > max2) ? max1 : max2;
   }
}

int main() {
   int a[100], n, min, max;
   printf("Enter number of elements: ");
   scanf("%d", &n);
   printf("Enter elements:\n");
   for (int i = 0; i < n; i++) scanf("%d", &a[i]);

   findMinMax(a, 0, n-1, &min, &max);
   printf("Minimum: %d\nMaximum: %d\n", min, max);
   return 0;
}
```

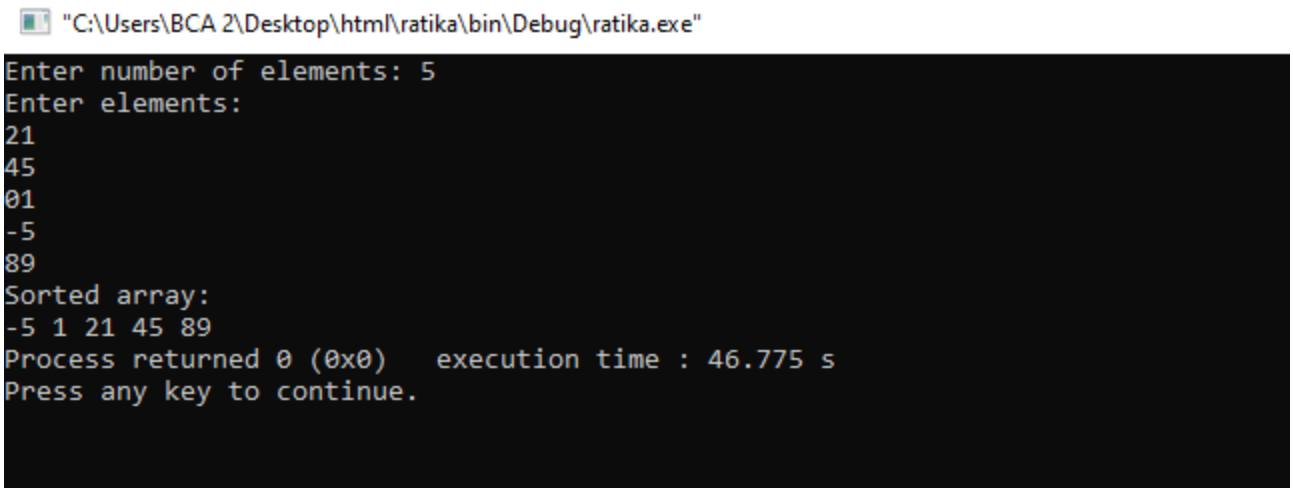
OUTPUT :



7. Write a test program to implement Divide and Conquer Strategy for Quick sort algorithm.

```
#include <stdio.h>
void quickSort(int a[], int low, int high) {
   if (low < high) {
      int pivot = a[high], i = low - 1;
      for (int j = low; j < high; j++)
         if (a[j] < pivot) {
            int t = a[++i]; a[i] = a[j]; a[j] = t;
         }
      int t = a[i+1]; a[i+1] = a[high]; a[high] = t;
      int pi = i + 1;

      quickSort(a, low, pi - 1);
      quickSort(a, pi + 1, high);
   }
}
int main() {
   int a[100], n;
   printf("Enter number of elements: ");
   scanf("%d", &n);
   printf("Enter elements:\n");
   for (int i = 0; i < n; i++) scanf("%d", &a[i]);

   quickSort(a, 0, n - 1);

   printf("Sorted array:\n");
   for (int i = 0; i < n; i++) printf("%d ", a[i]);
   return 0;
}
```

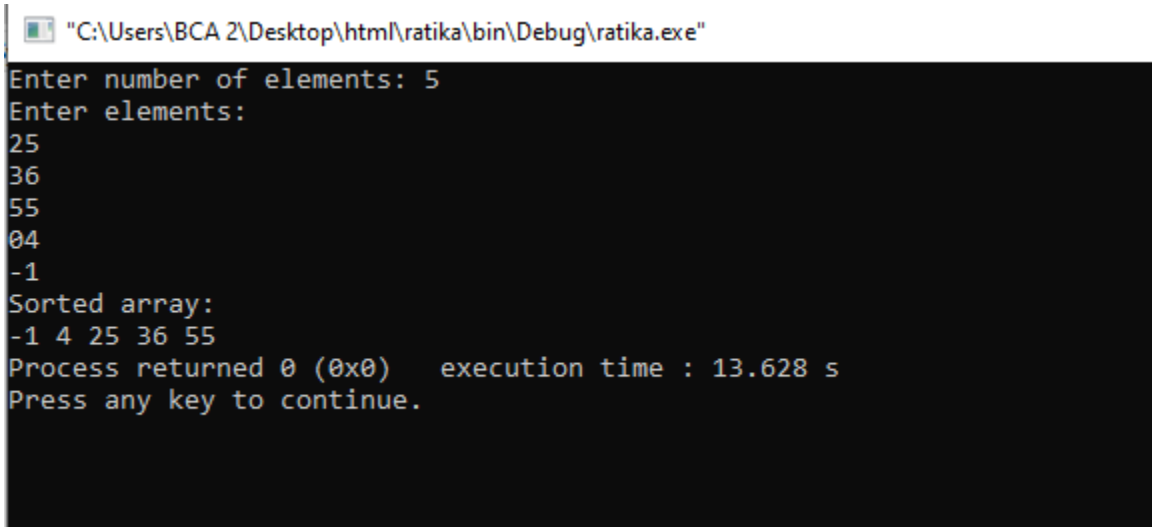
OUTPUT :



8. Write a program to implement Merge sort algorithm for sorting a list of integers in ascending order.

```
#include <stdio.h>
void merge(int a[], int l, int m, int r) {
   int n1 = m - l + 1, n2 = r - m;
   int L[n1], R[n2];
   for (int i = 0; i < n1; i++) L[i] = a[l + i];
   for (int i = 0; i < n2; i++) R[i] = a[m + 1 + i];

   int i = 0, j = 0, k = l;
   while (i < n1 && j < n2)
      a[k++] = (L[i] <= R[j]) ? L[i++] : R[j++];
   while (i < n1) a[k++] = L[i++];
   while (j < n2) a[k++] = R[j++];
}
void mergeSort(int a[], int l, int r) {
   if (l < r) {
      int m = l + (r - l) / 2;
      mergeSort(a, l, m);
      mergeSort(a, m + 1, r);
      merge(a, l, m, r);
   }
}
int main() {
   int n, a[100];
   printf("Enter number of elements: ");
   scanf("%d", &n);
   printf("Enter elements:\n");
   for (int i = 0; i < n; i++) scanf("%d", &a[i]);

   mergeSort(a, 0, n - 1);

   printf("Sorted array:\n");
   for (int i = 0; i < n; i++) printf("%d ", a[i]);
   return 0;
}
```

OUTPUT :


"C:\Users\BCA 2\Desktop\html\ratika\bin\Debug\ratika.exe"
Enter number of elements: 5
Enter elements:
25
36
55
04
-1
Sorted array:
-1 4 25 36 55
Process returned 0 (0x0)   execution time : 13.628 s
Press any key to continue.