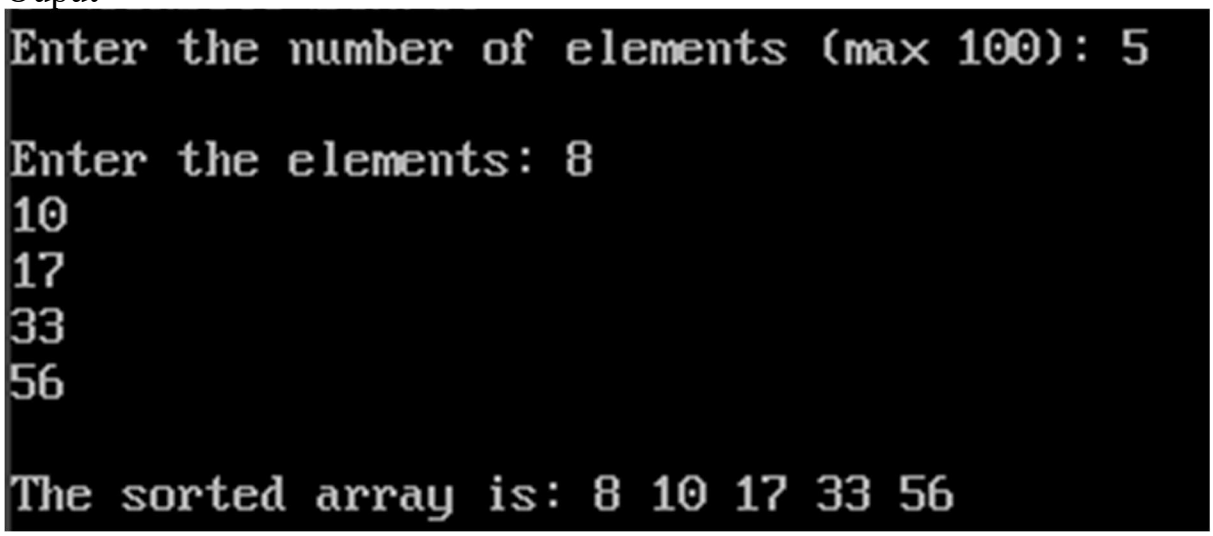
Name: Thakur Kuldeep Bhimsingh Batch: C-1 Roll No: 17

Code-

```
#include <stdio.h>
#define MAX_SIZE 100

void main() {
   int n, i, j, key;
   int a[MAX_SIZE];

   printf("Enter the number of elements (max %d): ", MAX_SIZE);
   scanf("%d", &n);

   if(n > MAX_SIZE) {
      printf("Error: number of elements exceeds the maximum allowed.\n");
      return;
   }
   printf("\nEnter the elements: ");
   for (i = 0; i < n; i++) {
      scanf("%d", &a[i]);
   }

   for (j = 1; j < n; j++) {
      key = a[j];
      i = j - 1;

      while (i >= 0 && a[i] > key) {
         a[i + 1] = a[i];
         i = i - 1;
      }
      a[i + 1] = key;
   }

   printf("\nThe sorted array is: ");
   for (j = 0; j < n; j++) {
      printf("%d ", a[j]);
   }
   printf("\n");
}
```

Ouput-


Enter the number of elements (max 100): 5
Enter the elements: 8
10
17
33
56
The sorted array is: 8 10 17 33 56

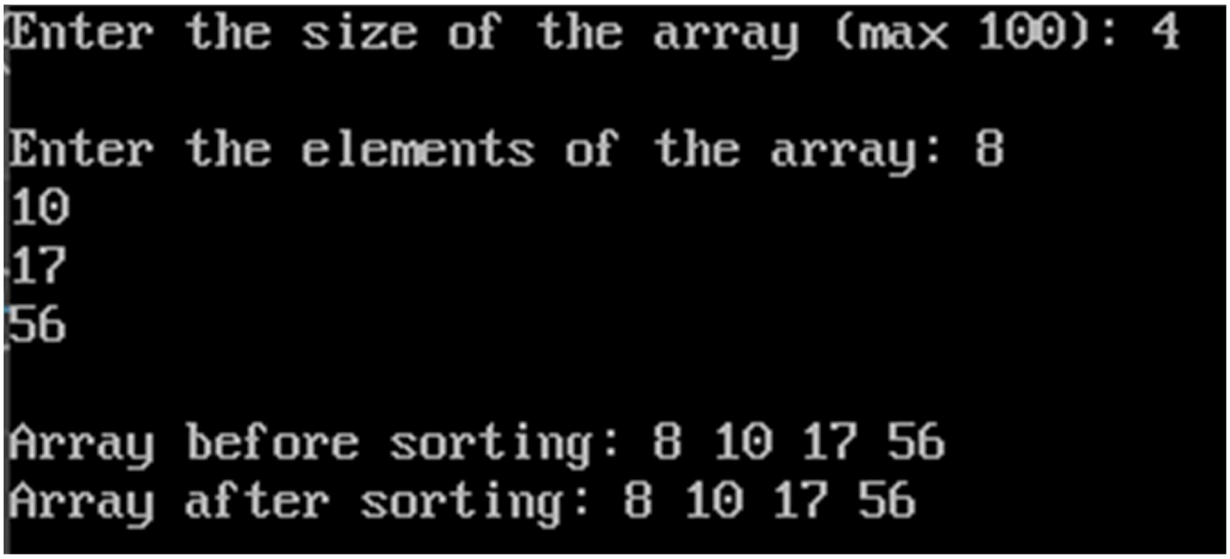
Name: Thakur Kuldeep Bhimsingh Batch: C-1 Roll No: 17

Code-

```
#include <stdio.h>
#include <conio.h>
#define MAX_SIZE 100

int main() {
   int n, arr[MAX_SIZE];
   int i, j, min, temp;

   clrscr();

   printf("Enter the size of the array (max %d): ", MAX_SIZE);
   scanf("%d", &n);

   if(n > MAX_SIZE) {
      printf("Size exceeds the maximum limit of %d.\n", MAX_SIZE);
      getch();
      return 1;
   }

   printf("\nEnter the elements of the array: ");
   for(i = 0; i < n; i++){
      scanf("%d", &arr[i]);
   }
```

```
    printf("\nArray before sorting: ");
    for(i = 0; i < n; i++){
        printf("%d ", arr[i]);
    }

    for(i = 0; i < n - 1; i++){
        min = i;
        for(j = i + 1; j < n; j++){
            if(arr[j] < arr[min]){
                min = j;
            }
        }
        if(min != i){
            temp = arr[i];
            arr[i] = arr[min];
            arr[min] = temp;
        }
    }

    printf("\nArray after sorting: ");
    for(i = 0; i < n; i++){
        printf("%d ", arr[i]);
    }

    getch();
    return 0;
}
```

Output-


Enter the size of the array (max 100): 4
Enter the elements of the array: 8
10
17
56
Array before sorting: 8 10 17 56
Array after sorting: 8 10 17 56

Name: Thakur Kuldeep Bhimsingh Batch: C-1 Roll No: 17

Code-

```
#include <stdio.h>
#include <stdlib.h>
#define MAX_SIZE 100

void merge_sort(int arr[], int p, int r);
void merge(int arr[], int p, int q, int r);
int main()
{
  int n, p, r, i;
  int arr[MAX_SIZE];\
 printf("Enter the number of elements (max %d): ",       MAX_SIZE);
scanf("%d", &n);
if (n > MAX_SIZE) {
   printf("Error: Number of elements exceeds the maximum allowed (%d).\n",
MAX_SIZE);
   return 1;
}

printf("Enter the elements of the array: ");
for (i = 0; i < n; i++) {
   scanf("%d", &arr[i]);
}

printf("\nThe elements of the array before sorting: ");
for (i = 0; i < n; i++) {
   printf("%d ", arr[i]);
```

```
}

p = 0;
r = n - 1;
merge_sort(arr, p, r);

printf("\nThe elements of the array after sorting: ");
for (i = 0; i < n; i++) {
    printf("%d ", arr[i]);
}
printf("\n");

return 0;

}
void merge_sort(int arr[], int p, int r)
{
  int q;
  if (p < r)
  {
   q = (p + r) / 2;
   merge_sort(arr, p, q);
   merge_sort(arr, q + 1, r);
   merge(arr, p, q, r);
  }
 }
void merge(int arr[], int p, int q, int r)
{
  int n1, n2;
 int left[MAX_SIZE], right[MAX_SIZE];
```

```
 int i, j, k;

n1 = q - p + 1;
n2 = r - q;

for (i = 0; i < n1; i++) {
   left[i] = arr[p + i];
}
for (j = 0; j < n2; j++) {
   right[j] = arr[q + 1 + j];
}

i = 0;
j = 0;
k = p;
while (i < n1 && j < n2) {
   if (left[i] <= right[j]) {
      arr[k] = left[i];
      i++;
   } else {
      arr[k] = right[j];
      j++;
   }
   k++;
}

while (i < n1) {
   arr[k] = left[i];
   i++;
   k++;
```

```
}


while (j < n2) {
    arr[k] = right[j];
    j++;
    k++;
}

}
```

Output-

```
Enter the number of elements (max 100): 6
Enter the elements of the array: 8
13
24
98
67
2

The elements of the array before sorting: 8 13 24 98 67 2
The elements of the array after sorting: 2 8 13 24 67 98
```

Name: Thakur Kuldeep Bhimsingh Batch: C-1 Roll No: 17

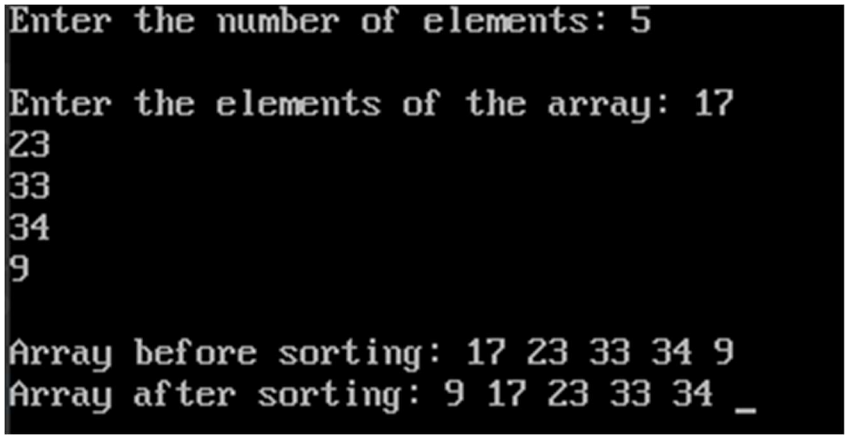
Code-

```
#include <stdio.h>
#include <conio.h>

void quicksort(int arr[], int p, int r);
int partition(int arr[], int p, int r);

int main() {
   int n, i;
   int arr[100];
   clrscr();

   printf("Enter the number of elements: ");
   scanf("%d", &n);

   printf("\nEnter the elements of the array: ");
   for (i = 0; i < n; i++) {
      scanf("%d", &arr[i]);
   }

   printf("\nArray before sorting: ");
   for (i = 0; i < n; i++) {
      printf("%d ", arr[i]);
   }

   quicksort(arr, 0, n - 1);
```

```
    printf("\nArray after sorting: ");
    for (i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }

    getch();
    return 0;
}

void quicksort(int arr[], int p, int r) {
    int q;
    if (p < r) {
        q = partition(arr, p, r);
        quicksort(arr, p, q - 1);
        quicksort(arr, q + 1, r);
    }
}

int partition(int arr[], int p, int r) {
    int x = arr[r];
    int i = p - 1;
    int j, temp;
    for (j = p; j < r; j++) {
        if (arr[j] <= x) {
            i++;
            temp = arr[i];
            arr[i] = arr[j];
            arr[j] = temp;
        }
```

```
    }
    temp = arr[i + 1];
    arr[i + 1] = arr[r];
    arr[r] = temp;
    return i + 1;
}
```

Output-



Name: Thakur Kuldeep Bhimsingh Batch: C-1 Roll No: 17

Code-

```
#include <stdio.h>
#include <stdlib.h>

typedef struct {
   double value, weight, cost;
} Item;

int compare(const void *a, const void *b) {
   double r1 = ((Item *)a)->cost;
   double r2 = ((Item *)b)->cost;
   return (r2 > r1) - (r1 > r2);
}

double fractionalKnapsack(int W, Item arr[], int n) {
   for (int i = 0; i < n; i++)
      arr[i].cost = arr[i].value / arr[i].weight;

   qsort(arr, n, sizeof(Item), compare);

   int i = 0;
   double total = 0.0;

   while (i < n) {
      if (arr[i].weight <= W) {
         W -= arr[i].weight;
         total += arr[i].value;
      } else {
         total += arr[i].value * ((double)W / arr[i].weight);
         break;
      }
      i++;
```

```
    }
    return total;
}

int main() {
    int n, W;
    printf("Enter number of items: ");
    scanf("%d", &n);

    Item arr[n];

    printf("Enter weight and value for each item:\n");
    for (int i = 0; i < n; i++)
        scanf("%lf %lf", &arr[i].weight, &arr[i].value);

    printf("Enter maximum capacity of knapsack: ");
    scanf("%d", &W);

    double maxValue = fractionalKnapsack(W, arr, n);
    printf("Maximum value in knapsack: %.2f\n", maxValue);

    return 0;
}
```

Output-

```
Enter number of items: 3
Enter weight and value for each item:
10 60
20 80
40 120
Enter maximum capacity of knapsack: 50
Maximum value in knapsack: 200.00
```

Name: Thakur Kuldeep Bhimsingh Batch: C-1 Roll No: 17

Code-

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#define INFINITY 9999
#define MAX 10

void dijkstra(int G[MAX][MAX], int n, int startnode);
int main() { int G[MAX][MAX], i, j, n, u;

clrscr();  /* Clear the screen (Turbo C specific) */
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);

getch();   /* Wait for key press before exiting (Turbo C specific) */
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;

int i, j; /* Declarations at the beginning of the function */
/* Create the cost matrix */
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];
   }
}

/* Initialize pred[], distance[] and visited[] */
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;

(count < n - 1) {
   mindistance = INFINITY;

   /* nextnode gives the node at minimum distance */
```

```
   for(i = 0; i < n; i++) {
      if(distance[i] < mindistance && !visited[i]) {
         mindistance = distance[i];
         nextnode = i;
      }
   }

   /* Check if a better path exists through nextnode */
   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++;
}

/* Print the path and distance of each node */
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);
```

```
        printf("\n");
    }
}

}
```

Output-

```
Enter no. of vertices:3

Enter the adjacency matrix:
0 8 4
8 0 2
4 2 0

Enter the starting node:0

Distance of node 1 = 6
Path = 1 <- 2 <- 0

Distance of node 2 = 4
Path = 2 <- 0
```

Name: Thakur Kuldeep Bhimsingh Batch: C-1 Roll No: 17

Code-

```
#include <stdio.h>
#include <string.h>
#include <conio.h>
#define MAX_SIZE 101 /* 100 characters max + 1 for extra index */

void LCS_Length(char X[], char Y[], int m, int n, int c[][MAX_SIZE], char b[][MAX_SIZE])
{ int i, j;

for (i = 0; i <= m; i++)
   c[i][0] = 0;
for (j = 0; j <= n; j++)
   c[0][j] = 0;

for (i = 1; i <= m; i++) {
   for (j = 1; j <= n; j++) {
      if (X[i - 1] == Y[j - 1]) {
         c[i][j] = c[i - 1][j - 1] + 1;
         b[i][j] = '\\';
      } else if (c[i - 1][j] >= c[i][j - 1]) {
         c[i][j] = c[i - 1][j];
         b[i][j] = '^';
      } else {
         c[i][j] = c[i][j - 1];
         b[i][j] = '<';
      }
   }
}

}
```

```
void Print_LCS(char b[][MAX_SIZE], char X[], int i, int j)
{
 if (i == 0 || j == 0) return;
if (b[i][j] == '\')
{
  Print_LCS(b, X, i - 1, j - 1);
   printf("%c", X[i - 1]);
}
 else if (b[i][j] == '^')
{
 Print_LCS(b, X, i - 1, j);
}
else
{
 Print_LCS(b, X, i, j - 1);
}
}

int main()
{
  char X[100], Y[100];
  int m, n; int c[MAX_SIZE][MAX_SIZE];
  char b[MAX_SIZE][MAX_SIZE];

clrscr();

printf("Enter the first string: ");
scanf("%s", X);

printf("Enter the second string: ");
scanf("%s", Y);
```

```
m = strlen(X);
n = strlen(Y);

LCS_Length(X, Y, m, n, c, b);

printf("The Longest Common Subsequence (LCS) is: ");
Print_LCS(b, X, m, n);
printf("\n");

getch();
return 0;

}
```

Output-

```
Enter the first string: ADCDADCBADC
Enter the second string: ABCD
The Longest Common Subsequence (LCS) is: ACD
```

Name: Thakur Kuldeep Bhimsingh Batch: C-1 Roll No: 17

Code-

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
int a[30], count = 0;

int place(int pos)
{
  int i; for(i = 1; i < pos; i++)
  {
     if ((a[i] == a[pos]) || (abs(a[i] - a[pos]) == abs(i - pos)))
     return 0;
   }
   return 1;
}
void print_sol(int n)
{
  int i, j;
  count++;
  printf("\n\nSolution #%d:\n", count);
  for(i = 1; i <= n; i++)
 {
    for(j = 1; j <= n; j++)
    { if(a[i] == j) printf("Q\t");
       else printf("*\t");
  }
  printf("\n");
  }
}

void queen(int n)
{
```

```
  int k;
  k = 1;
  a[k] = 0;

  while(k != 0)
  {
     a[k] = a[k] + 1;
     while((a[k] <= n) && !place(k))
      {
        a[k] = a[k] + 1;
      }
      if(a[k] <= n)
      {
          if(k == n) print_sol(n);
      }
      else
      {
        k++; a[k] = 0;
      }
   }
    else
   {
     k--;
   }
  }
}

void main()
 {
    int n;
    clrscr();
```

```
    printf("Enter the number of Queens\n");
    scanf("%d", &n);
    queen(n);
    printf("\nTotal solutions = %d", count);
    getch();
}
```

Output-

```
Enter the number of Queens
2

Total solutions = 0
```

```
Enter the number of Queens
4


Solution #1:
*       Q       *       *
*       *       *       Q
Q       *       *       *
*       *       Q       *


Solution #2:
*       *       Q       *
Q       *       *       *
*       *       *       Q
*       Q       *       *

Total solutions = 2
```

Name: Thakur Kuldeep Bhimsingh Batch: C-1 Roll No: 17

Code-

```
#include <stdio.h>
#include <string.h>
int main()
{
  char txt[] = "analysisofalgorithm";
  char pat[] = "a";
  int M = strlen(pat);
  int N = strlen(txt);
  int i, j;
  for (i = 0; i <= N - M; i++) {
    for (j = 0; j < M; j++) {
        if (txt[i + j] != pat[j]) {
            break;
        }
    }
    if (j == M) {
        printf("Pattern matches at index %d\n", i);
    }
 }
 return 0;
}
```

Output-

```
Pattern matches at index 0
Pattern matches at index 2
Pattern matches at index 10
```

Name: Thakur Kuldeep Bhimsingh Batch: C-1 Roll No: 17

Code-

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
#define tonum(c) ((c) >= 'A' && (c) <= 'Z' ? (c) - 'A' : (c) - 'a' + 26)

int mod(int a, int p, int m)
{
  int sqr;
  if (p == 0) return 1;
    sqr = mod(a, p / 2, m) % m;
    sqr = (sqr * sqr) % m;
    if (p & 1) return ((a % m) * sqr) % m;
    else return sqr;
   }
    int RabinKarpMatch(char *T, char *P, int d, int q)
    {
            int i, j, p, t, n, m, h, found;
             n = strlen(T);
            m = strlen(P);
            h = mod(d, m - 1, q);
            p = 0;
            t = 0;

      for (i = 0; i < m; i++) {
         p = (d * p + tonum(P[i])) % q;
         t = (d * t + tonum(T[i])) % q;
      }
```

```
    for (i = 0; i <= n - m; i++) {
        if (p == t) {
            found = 1;
            for (j = 0; j < m; j++) {
                if (P[j] != T[i + j]) {
                    found = 0;
                    break;
                }
            }
            if (found)
                return i;
        }
        else {
            t = (d * (t - ((tonum(T[i]) * h) % q)) + tonum(T[i + m])) % q;
            if (t < 0)
                t = t + q;
        }
    }
    return -1;

}
void main() { char str[100], p[100]; int ans, q;
clrscr();

printf("\n Enter String: ");
gets(str);

printf("\n Enter Pattern you want to search in the string: ");
gets(p);
```

```
printf("\n Enter value of q: ");
scanf("%d", &q);

ans = RabinKarpMatch(str, p, 10, q);

if (ans == -1)
   printf("\n Pattern is not found.");
else
   printf("\n Pattern is found at displacement: %d", ans);
getch();

}
```

Output-

```
Enter String: Pattern Matching Algorithm

Enter Pattern you want to search in the string: Match

Enter value of q: 101

Pattern is found at displacement: 8
```

Name: Thakur Kuldeep Bhimsingh Batch: C-1 Roll No: 17