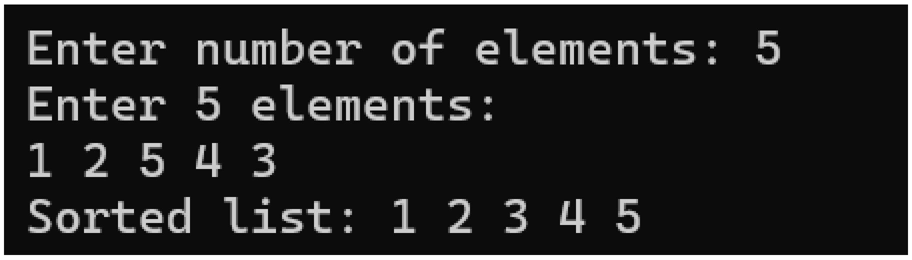
Program -1

```
#include <stdio.h>
void sort(int arr[], int n) {
    int i, j, temp;
    for(i = 0; i < n-1; i++) {
        for(j = 0; j < n-i-1; j++) {
            if(arr[j] > arr[j+1]) {
                temp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = temp;
            }
        }
    }
}
int main() {
    int arr[100], n, i;
    printf("Enter number of elements: ");
    scanf("%d", &n);
    printf("Enter %d elements:\n", n);
    for(i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }
    sort(arr, n);
    printf("Sorted list: ");
    for(i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    return 0;
}
```

Output:

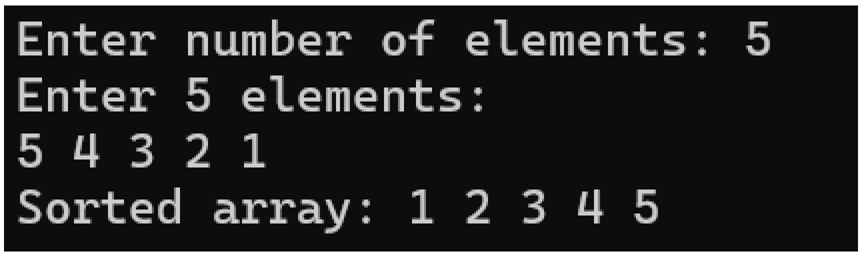


```
Enter number of elements: 5
Enter 5 elements:
1 2 5 4 3
Sorted list: 1 2 3 4 5
```

Program -2

```
#include <stdio.h>
void selectionSort(int arr[], int n) {
    int i, j, minIndex, temp;
    for (i = 0; i < n - 1; i++) {
        minIndex = i;
        for (j = i + 1; j < n; j++) {
            if (arr[j] < arr[minIndex]) {
                minIndex = j;
            }
        }
        temp = arr[i];
        arr[i] = arr[minIndex];
        arr[minIndex] = temp;
    }
}
int main() {
    int n, i;
    int arr[100];
    printf("Enter number of elements: ");
    scanf("%d", &n);
    printf("Enter %d elements:\n", n);
    for (i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }
    selectionSort(arr, n);
    printf("Sorted array: ");
    for (i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    return 0;
}
```

Output:

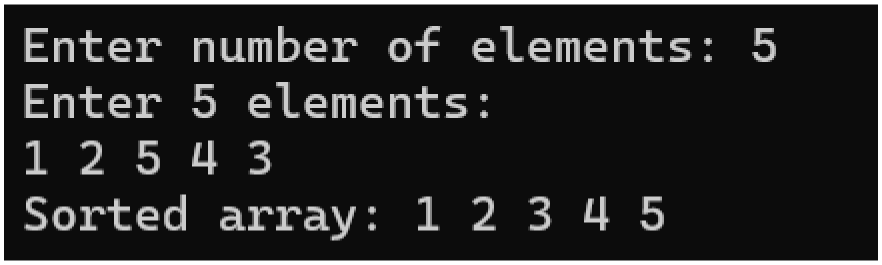


```
Enter number of elements: 5
Enter 5 elements:
5 4 3 2 1
Sorted array: 1 2 3 4 5
```

Program -3

```
#include <stdio.h>
void bubbleSortOptimized(int arr[], int n) {
    int i, j, temp;
    int swapped;
    for (i = 0; i < n - 1; i++) {
        swapped = 0;
        for (j = 0; j < n - i - 1; j++) {
            if (arr[j] > arr[j + 1]) {
                temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
                swapped = 1;
            }
        }
        if (swapped == 0)
            break;
    }
}
int main() {
    int n, i;
    int arr[100];
    printf("Enter number of elements: ");
    scanf("%d", &n);
    printf("Enter %d elements:\n", n);
    for (i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }
    bubbleSortOptimized(arr, n);
    printf("Sorted array: ");
    for (i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    return 0;
}
```

Output:

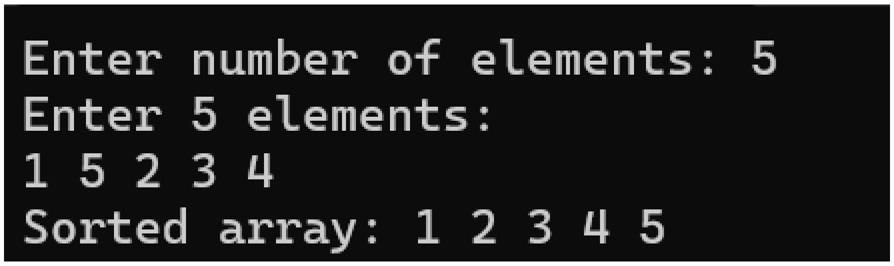


```
Enter number of elements: 5
Enter 5 elements:
1 2 5 4 3
Sorted array: 1 2 3 4 5
```

Program -4

```
#include <stdio.h>
void insertionSort(int arr[], int n) {
    int i, key, j;
    for (i = 1; i < n; i++) {
        key = arr[i];
        j = i - 1;
        while (j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
            j--;
        }
        arr[j + 1] = key;
    }
}
int main() {
    int n, i;
    int arr[100];
    printf("Enter number of elements: ");
    scanf("%d", &n);
    printf("Enter %d elements:\n", n);
    for (i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }
    insertionSort(arr, n);
    printf("Sorted array: ");
    for (i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    return 0;
}
```

Output:



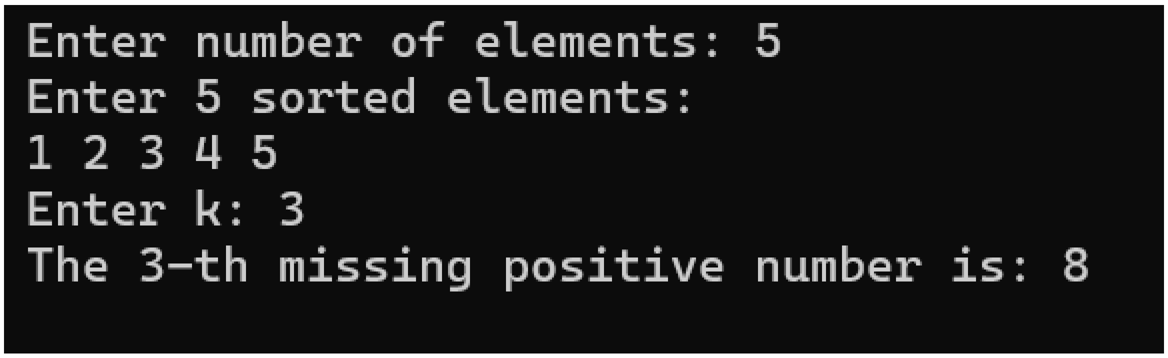
```
Enter number of elements: 5
Enter 5 elements:
1 5 2 3 4
Sorted array: 1 2 3 4 5
```

Program -5:

```
#include <stdio.h>
int findKthMissing(int arr[], int n, int k) {
    int i;
    for (i = 0; i < n; i++) {
        int missing = arr[i] - (i + 1);
        if (missing >= k) {
            return k + i;
        }
    }
    return k + n;
}
```

```
int main() {
    int n, k, i;
    int arr[100];
    printf("Enter number of elements: ");
    scanf("%d", &n);
    printf("Enter %d sorted elements:\n", n);
    for (i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }
    printf("Enter k: ");
    scanf("%d", &k);
    int result = findKthMissing(arr, n, k);
    printf("The %d-th missing positive number is: %d\n", k, result);
    return 0;
}
```

Output:

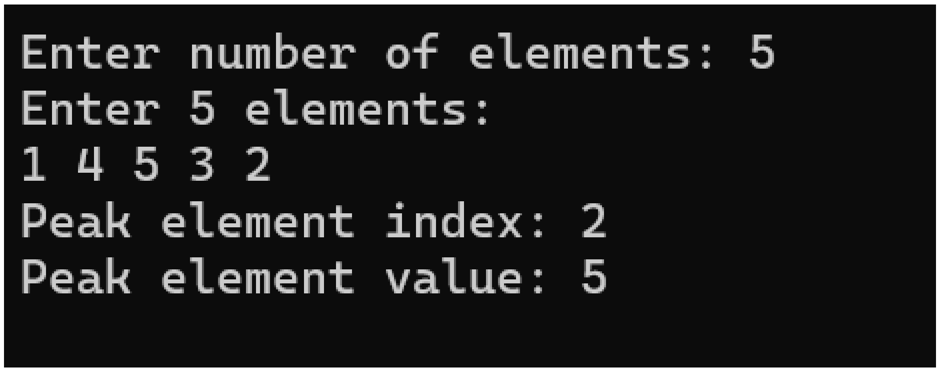


```
Enter number of elements: 5
Enter 5 sorted elements:
1 2 3 4 5
Enter k: 3
The 3-th missing positive number is: 8
```

Program -6

```
#include <stdio.h>
int findPeakElement(int nums[], int n) {
    int left = 0, right = n - 1;
    while (left < right) {
        int mid = (left + right) / 2;
        if (nums[mid] < nums[mid + 1]) {
            left = mid + 1;
        }
        else {
            right = mid;
        }
    }
    return left;
}
int main() {
    int n, i;
    int nums[100];
    printf("Enter number of elements: ");
    scanf("%d", &n);
    printf("Enter %d elements:\n", n);
    for (i = 0; i < n; i++) {
        scanf("%d", &nums[i]);
    }
    int index = findPeakElement(nums, n);
    printf("Peak element index: %d\n", index);
    printf("Peak element value: %d\n", nums[index]);
    return 0;
}
```

Output:

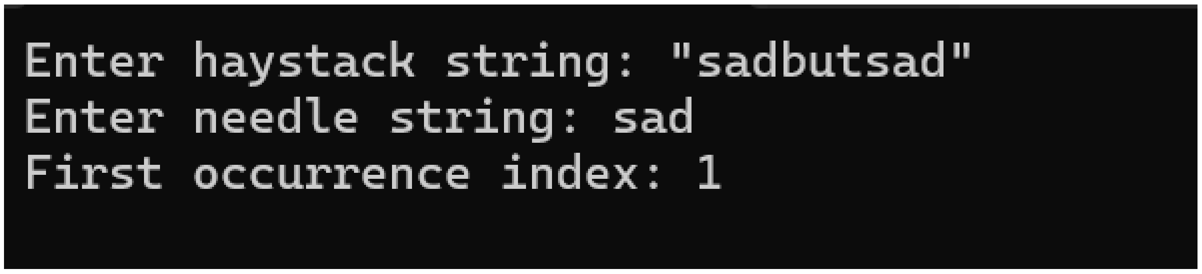


```
Enter number of elements: 5
Enter 5 elements:
1 4 5 3 2
Peak element index: 2
Peak element value: 5
```

Program-7

```
#include <stdio.h>
#include <string.h>
int strStr(char haystack[], char needle[]) {
    int hLen = strlen(haystack);
    int nLen = strlen(needle);
    if (nLen == 0)
        return 0;
    for (int i = 0; i <= hLen - nLen; i++) {
        int j;
        for (j = 0; j < nLen; j++) {
            if (haystack[i + j] != needle[j])
            break;
        }
        if (j == nLen)
            return i;
    }
    return -1;
}
int main() {
    char haystack[100], needle[100];
    printf("Enter haystack string: ");
    scanf("%s", haystack);
    printf("Enter needle string: ");
    scanf("%s", needle);
    int index = strStr(haystack, needle);
    printf("First occurrence index: %d\n", index);
    return 0;
}
```

Output:

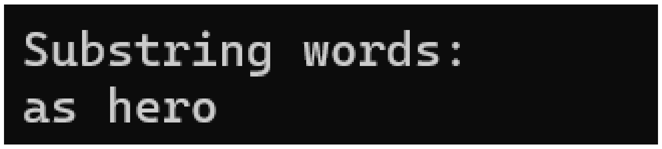


```
Enter haystack string: "sadbutsad"
Enter needle string: sad
First occurrence index: 1
```

Program-8

```
#include <stdio.h>
#include <string.h>
int main() {
    char words[10][50] = {"mass", "as", "hero", "superhero"};
    int n = 4;
    printf("Substring words:\n");
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            if (i != j) {
                if (strstr(words[j], words[i]) != NULL) {
                    printf("%s ", words[i]);
                    break;
                }
            }
        }
    }
    return 0;
}
```

Output:

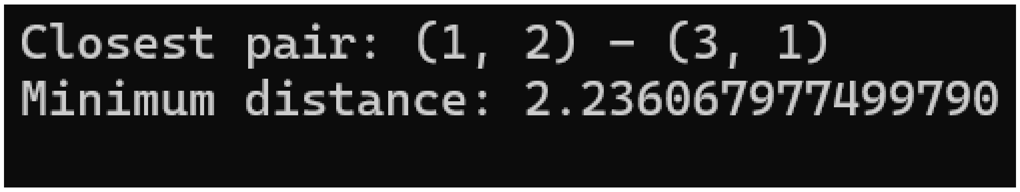


```
Substring words:
as hero
```

Program-9

```
#include <stdio.h>
#include <math.h>
#include <float.h>
typedef struct {
    double x, y;
} Point;
double distance(Point a, Point b) {
    return sqrt((a.x - b.x)*(a.x - b.x) + (a.y - b.y)*(a.y - b.y));
}
int main() {
    Point points[] = {{1, 2}, {4, 5}, {7, 8}, {3, 1}};
    int n = 4;
    double minDist = DBL_MAX;
    Point p1, p2;
    for (int i = 0; i < n; i++) {
        for (int j = i + 1; j < n; j++) {
            double d = distance(points[i], points[j]);
            if (d < minDist) {
                minDist = d;
                p1 = points[i];
                p2 = points[j];
            }
        }
    }
    printf("Closest pair: (%.0f, %.0f) - (%.0f, %.0f)\n", p1.x, p1.y, p2.x, p2.y);
    printf("Minimum distance: %.15f\n", minDist);
    return 0;
}
```

Output:

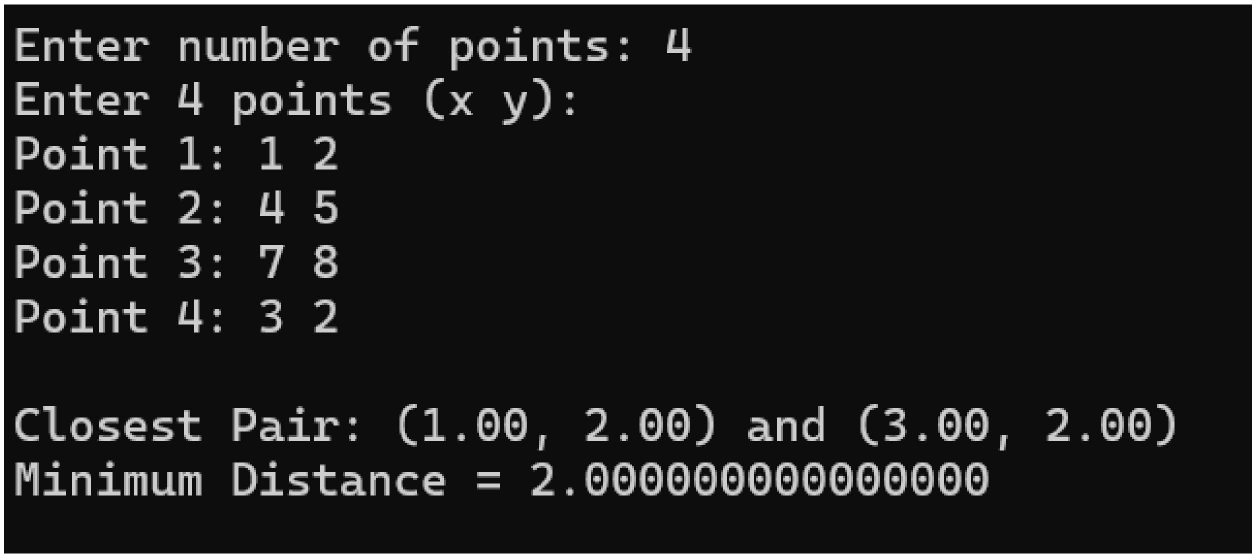


```
Closest pair: (1, 2) - (3, 1)
Minimum distance: 2.236067977499790
```

Program:10

```
#include <stdio.h>
#include <math.h>
#include <float.h>
typedef struct {
    double x, y;
} Point;
double distance(Point a, Point b) {
    return sqrt((a.x - b.x)*(a.x - b.x) + (a.y - b.y)*(a.y - b.y));
}
void closestPair(Point pts[], int n) {
    double minDist = DBL_MAX;
    Point p1, p2;
    for (int i = 0; i < n; i++) {
        for (int j = i + 1; j < n; j++) {
            double d = distance(pts[i], pts[j]);
            if (d < minDist) {
                minDist = d;
                p1 = pts[i];
                p2 = pts[j];
            }
        }
    }
    printf("\nClosest Pair: (%.2f, %.2f) and (%.2f, %.2f)\n",
            p1.x, p1.y, p2.x, p2.y);
    printf("Minimum Distance = %.15f\n", minDist);
}
int main() {
    int n;
    printf("Enter number of points: ");
    scanf("%d", &n);
    Point points[n];
    printf("Enter %d points (x y):\n", n);
    for (int i = 0; i < n; i++) {
        printf("Point %d: ", i + 1);
        scanf("%lf %lf", &points[i].x, &points[i].y);
    }
    closestPair(points, n);
    return 0;
}
```

Output:



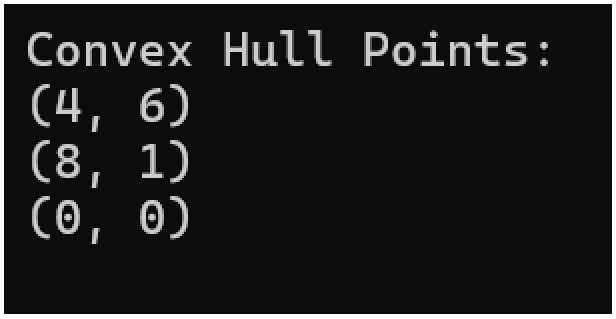
```
Enter number of points: 4
Enter 4 points (x y):
Point 1: 1 2
Point 2: 4 5
Point 3: 7 8
Point 4: 3 2

Closest Pair: (1.00, 2.00) and (3.00, 2.00)
Minimum Distance = 2.000000000000000
```

Program-11

```
#include <stdio.h>
typedef struct {
   int x, y;
} Point;
int direction(Point a, Point b, Point c) {
   return (b.x - a.x) * (c.y - a.y) -
          (b.y - a.y) * (c.x - a.x);
}
int main() {
   Point pts[] = {{1, 1}, {4, 6}, {8, 1}, {0, 0}, {3, 3}};
   int n = 5;
```

```
Point hull[20];
int hullCount = 0;
for (int i = 0; i < n; i++) {
    for (int j = i + 1; j < n; j++) {
        int left = 0, right = 0;
        for (int k = 0; k < n; k++) {
            if (k == i || k == j) continue;
            int d = direction(pts[i], pts[j], pts[k]);
            if (d > 0) left++;
            else if (d < 0) right++;
        }
        if (left == 0 || right == 0) {
            hull[hullCount++] = pts[i];
            hull[hullCount++] = pts[j];
        }
    }
}
Point unique[20];
int uCount = 0;
for (int i = 0; i < hullCount; i++) {
    int exists = 0;
    for (int j = 0; j < uCount; j++) {
        if (hull[i].x == unique[j].x && hull[i].y == unique[j].y)
            exists = 1;
    }
    if (!exists)
        unique[uCount++] = hull[i];
    }

    printf("Convex Hull Points:\n");
    for (int i = 0; i < uCount; i++) {
        printf("(%d, %d)\n", unique[i].x, unique[i].y);
    }
    return 0;
}
```

Output:



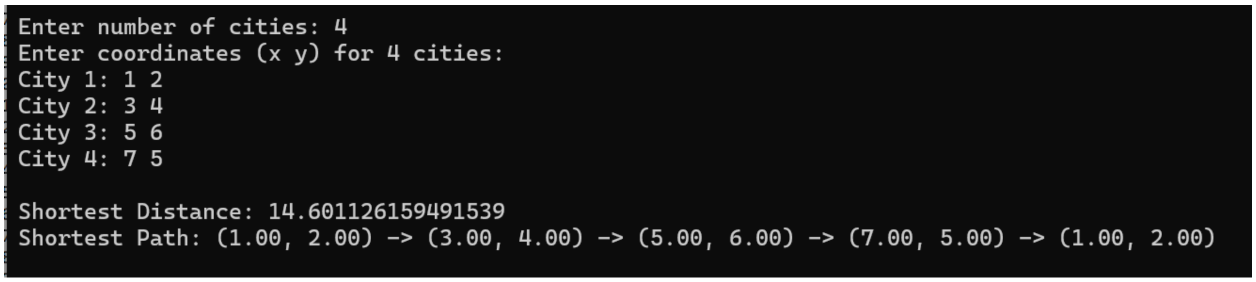
```
Convex Hull Points:
(4, 6)
(8, 1)
(0, 0)
```

Program-12

```
#include <stdio.h>
#include <math.h>
#include <float.h>

#define MAX 12
typedef struct {
    double x, y;
} Point;
```

```
double distance(Point a, Point b) {
    return sqrt((a.x - b.x)*(a.x - b.x) + (a.y - b.y)*(a.y - b.y));
}
void swap(Point *a, Point *b) {
    Point temp = *a;
    *a = *b;
    *b = temp;
}
double minDist = DBL_MAX;
Point bestPath[MAX];
int bestLen;
void permute(Point arr[], int l, int r, Point startCity) {
    if (l == r) {
        double total = 0.0;
        Point prev = startCity;
        for (int i = 0; i <= r; i++) {
            total += distance(prev, arr[i]);
            prev = arr[i];
        }
        total += distance(prev, startCity); // return to start
        if (total < minDist) {
            minDist = total;
            bestLen = r + 1;
            bestPath[0] = startCity;
            for (int i = 0; i <= r; i++) {
                bestPath[i+1] = arr[i];
            }
            bestPath[r+2] = startCity;
        }
        return;
    }
    for (int i = l; i <= r; i++) {
        swap(&arr[l], &arr[i]);
        permute(arr, l + 1, r, startCity);
        swap(&arr[l], &arr[i]); // backtrack
    }
}
int main() {
    int n;
    printf("Enter number of cities: ");
    scanf("%d", &n);
    if (n < 2) {
        printf("At least 2 cities required.\n");
        return 0;
    }
    Point cities[MAX];
    printf("Enter coordinates (x y) for %d cities:\n", n);
    for (int i = 0; i < n; i++) {
        printf("City %d: ", i + 1);
        scanf("%lf %lf", &cities[i].x, &cities[i].y);
    }
    Point startCity = cities[0];
    permute(cities + 1, 0, n - 2, startCity);
    printf("\nShortest Distance: %.15f\n", minDist);
    printf("Shortest Path: ");
    for (int i = 0; i < bestLen + 2; i++) {
        printf("(%.2f, %.2f)", bestPath[i].x, bestPath[i].y);
        if (i < bestLen + 1) printf(" -> ");
    }
    printf("\n");
    return 0;
}
```

Output:

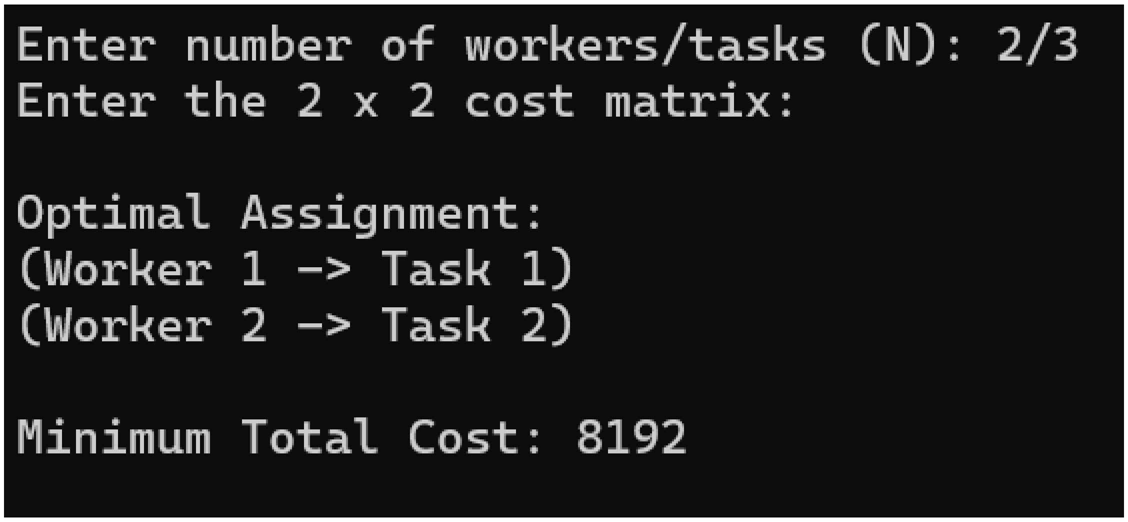


```
Enter number of cities: 4
Enter coordinates (x y) for 4 cities:
City 1: 1 2
City 2: 3 4
City 3: 5 6
City 4: 7 5

Shortest Distance: 14.601126159491539
Shortest Path: (1.00, 2.00) -> (3.00, 4.00) -> (5.00, 6.00) -> (7.00, 5.00) -> (1.00, 2.00)
```

Program-13

```
#include <stdio.h>
int minCost = 999999;
int bestAssignment[20];
void swap(int *a, int *b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}
int total_cost(int assignment[], int cost[][20], int n) {
    int sum = 0;
    for (int i = 0; i < n; i++)
        sum += cost[i][assignment[i]];
    return sum;
}
void permute(int assignment[], int l, int r, int cost[][20], int n) {
    if (l == r) {
        int currCost = total_cost(assignment, cost, n);
        if (currCost < minCost) {
            minCost = currCost;
            for (int i = 0; i < n; i++)
                bestAssignment[i] = assignment[i];
        }
        return;
    }
    for (int i = l; i <= r; i++) {
        swap(&assignment[l], &assignment[i]);
        permute(assignment, l + 1, r, cost, n);
        swap(&assignment[l], &assignment[i]);
    }
}
int main() {
    int n;
    int cost[20][20];
    int assignment[20];
    printf("Enter number of workers/tasks (N): ");
    scanf("%d", &n);
    printf("Enter the %d x %d cost matrix:\n", n, n);
    for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++)
            scanf("%d", &cost[i][j]);
    for (int i = 0; i < n; i++)
        assignment[i] = i;
    permute(assignment, 0, n - 1, cost, n);
    printf("\nOptimal Assignment:\n");
    for (int i = 0; i < n; i++)
        printf("(Worker %d -> Task %d)\n", i + 1, bestAssignment[i] + 1);
    printf("\nMinimum Total Cost: %d\n", minCost);
    return 0;
}
```

Output:

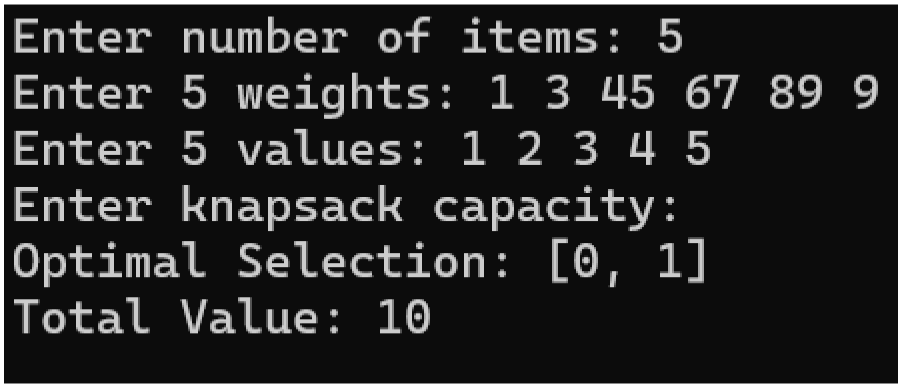


```
Enter number of workers/tasks (N): 2/3
Enter the 2 x 2 cost matrix:

Optimal Assignment:
(Worker 1 -> Task 1)
(Worker 2 -> Task 2)

Minimum Total Cost: 8192
```

Program:14

```
#include <stdio.h>
int bestValue = 0;
int bestSet[20], bestSize = 0;
int total_value(int items[], int size, int values[]) {
    int sum = 0;
    for (int i = 0; i < size; i++)
        sum += values[items[i]];
    return sum;
}
int is_feasible(int items[], int size, int weights[], int capacity) {
    int w = 0;
    for (int i = 0; i < size; i++)
        w += weights[items[i]];
    return w <= capacity;
}
void exhaustive(int index, int n, int weights[], int values[], int capacity,
                int currentSet[], int currentSize) {
    if (index == n) {
        if (is_feasible(currentSet, currentSize, weights, capacity)) {
            int val = total_value(currentSet, currentSize, values);
            if (val > bestValue) {
                bestValue = val;
                bestSize = currentSize;
                for (int i = 0; i < currentSize; i++)
                    bestSet[i] = currentSet[i];
            }
        }
        return;
    }
    currentSet[currentSize] = index;
    exhaustive(index + 1, n, weights, values, capacity, currentSet, currentSize + 1);
    exhaustive(index + 1, n, weights, values, capacity, currentSet, currentSize);
}
int main() {
    int n, capacity;
    int weights[20], values[20];
    int currentSet[20];
    printf("Enter number of items: ");
    scanf("%d", &n);
    printf("Enter %d weights: ", n);
    for (int i = 0; i < n; i++)
        scanf("%d", &weights[i]);
    printf("Enter %d values: ", n);
    for (int i = 0; i < n; i++)
        scanf("%d", &values[i]);
    printf("Enter knapsack capacity: ");
    scanf("%d", &capacity);
    exhaustive(0, n, weights, values, capacity, currentSet, 0);
    printf("\nOptimal Selection: [");
    for (int i = 0; i < bestSize; i++) {
        printf("%d", bestSet[i]);
        if (i < bestSize - 1) printf(", ");
    }
    printf("]\n");
    printf("Total Value: %d\n", bestValue);
    return 0;
}
```

Output:



```
Enter number of items: 5
Enter 5 weights: 1 3 45 67 89 9
Enter 5 values: 1 2 3 4 5
Enter knapsack capacity:
Optimal Selection: [0, 1]
Total Value: 10
```