

Day -4

Assignment

Menda Mani Sai

192111399

1. Write a program to inset a number in a list.

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

```
int main() {
```

```
    int originalArray[100];
```

```
    int newArray[101];
```

```
    int n, num, pos, i;
```

```
    printf("Enter the number of elements in the array: ");
```

```
    scanf("%d", &n);
```

```
    printf("Enter %d elements:\n", n);
```

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

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

```
    }
```

```
    printf("Enter the number to insert: ");
```

```
    scanf("%d", &num);
```

```
    printf("Enter the position to insert (0-%d): ", n);
```

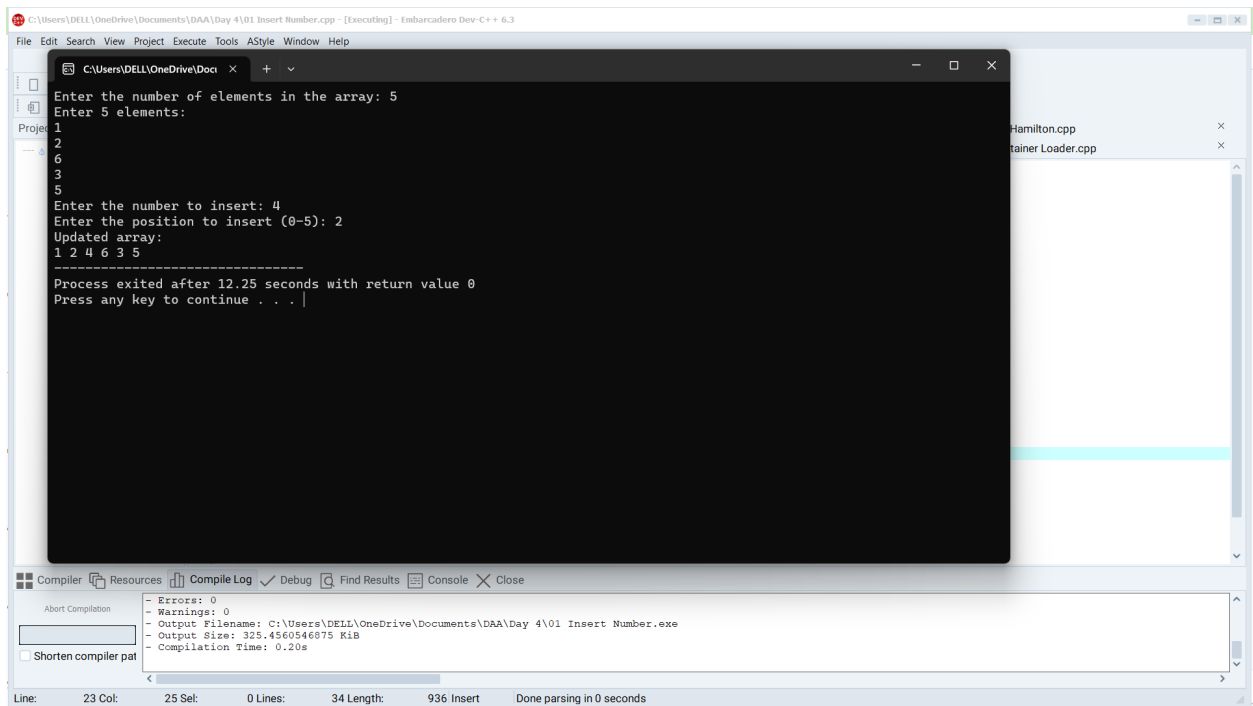
```
    scanf("%d", &pos);
```

```
    if (pos < 0 || pos > n) {
```

```
        printf("Invalid position. Position should be between 0 and %d\n", n);
```

```
    return 1;
}
for (i = 0; i < pos; i++) {
    newArray[i] = originalArray[i];
}
newArray[pos] = num;
for (i = pos; i < n; i++) {
    newArray[i + 1] = originalArray[i];
}
n++;
printf("Updated array:\n");
for (i = 0; i < n; i++) {
    printf("%d ", newArray[i]);
}
return 0;
}
```

OUTPUT:



```
C:\Users\DELL\OneDrive\Documents\DAA\Day 4\01 Insert Number.cpp - [Executing] - Embarcadero Dev-C++ 6.3
File Edit Search View Project Execute Tools AStyle Window Help
C:\Users\DELL\OneDrive\Documents\DAA\Day 4\01 Insert Number.cpp
Enter the number of elements in the array: 5
Enter 5 elements:
1
2
6
3
5
Enter the number to insert: 4
Enter the position to insert (0-5): 2
Updated array:
1 2 4 6 3 5
-----
Process exited after 12.25 seconds with return value 0
Press any key to continue . . . |
Compiler Resources Compile Log Debug Find Results Console Close
Abort Compilation
Errors: 0
Warnings: 0
Output Filename: C:\Users\DELL\OneDrive\Documents\DAA\Day 4\01 Insert Number.exe
Output Size: 325.4560546875 KiB
Compilation Time: 0.20s
Shorten compiler path
Line: 23 Col: 25 Sel: 0 Lines: 34 Length: 936 Insert Done parsing in 0 seconds
```

2. Write a program to perform sum of subsets problem using backtracking.

```
#include <stdio.h>

#define MAX_SIZE 100

int set[MAX_SIZE];

int solution[MAX_SIZE];

int n, targetSum;

void subsetSum(int index, int currentSum, int size) {

    if (currentSum == targetSum) {

        printf("Subset: ");

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

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

        }

    }

}
```

```

    printf("\n");

    return;

}

if (currentSum > targetSum || index >= n) {

    return;

}

solution[size] = set[index];

subsetSum(index + 1, currentSum + set[index], size + 1);

subsetSum(index + 1, currentSum, size);

}

```

```

int main() {

    printf("Enter the number of elements in the set: ");

    scanf("%d", &n);

    printf("Enter the elements of the set:\n");

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

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

    }

    printf("Enter the target sum: ");

    scanf("%d", &targetSum);

    printf("Subsets with the sum %d:\n", targetSum);

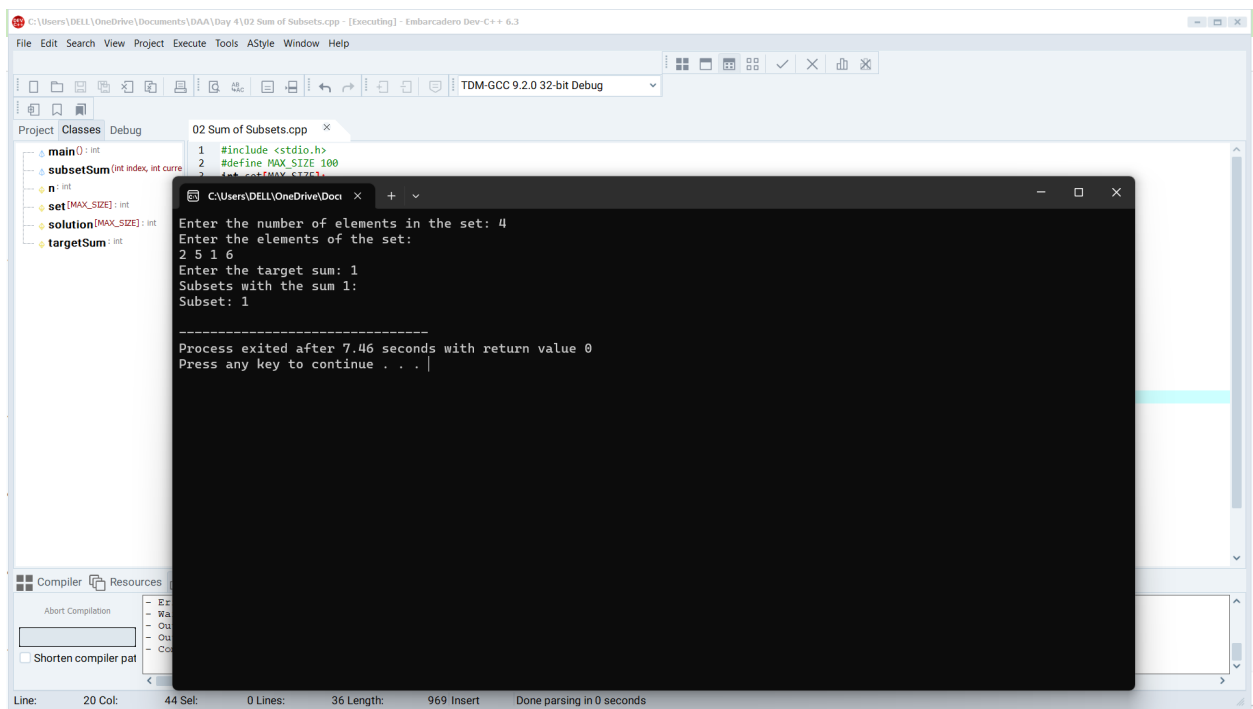
    subsetSum(0, 0, 0);

    return 0;

}

```

OUTPUT:



```
02 Sum of Subsets.cpp
1 #include <stdio.h>
2 #define MAX_SIZE 100
3 int set[MAX_SIZE];
4 int n;
5 int solution[MAX_SIZE];
6 int targetSum;

main() {
    subsetSum(0, 0, 0);
}

subsetSum(int index, int currentSum, int currentSet[]) {
    if (index == n) {
        if (currentSum == targetSum) {
            printf("Subsets with the sum %d:\n", targetSum);
            for (int i = 0; i < n; i++) {
                printf("%d ", currentSet[i]);
            }
            printf("\n");
        }
    } else {
        currentSet[index] = currentSet[index-1];
        subsetSum(index+1, currentSum, currentSet);
        currentSet[index] = currentSet[index-1] + 1;
        subsetSum(index+1, currentSum, currentSet);
    }
}

Enter the number of elements in the set: 4
Enter the elements of the set:
2 5 1 6
Enter the target sum: 1
Subsets with the sum 1:
Subset: 1

Process exited after 7.46 seconds with return value 0
Press any key to continue . . .
```

3. Write a program to perform graph coloring problem using backtracking.

```
#include <stdio.h>

#include <stdbool.h>

#define MAX_VERTICES 100

int graph[MAX_VERTICES][MAX_VERTICES];

int colors[MAX_VERTICES];

int numVertices, numColors;

bool isSafe(int v, int c) {
    for (int i = 0; i < numVertices; i++) {
        if (graph[v][i] && colors[i] == c) {
            return false;
        }
    }
}
```

```

    }

    return true;
}

bool graphColoring(int v) {
    if (v == numVertices) {
        return true;
    }

    for (int c = 1; c <= numColors; c++) {
        if (isSafe(v, c)) {
            colors[v] = c;

            if (graphColoring(v + 1)) {
                return true;
            }

            colors[v] = 0;
        }
    }

    return false;
}

int main() {
    printf("Enter the number of vertices: ");

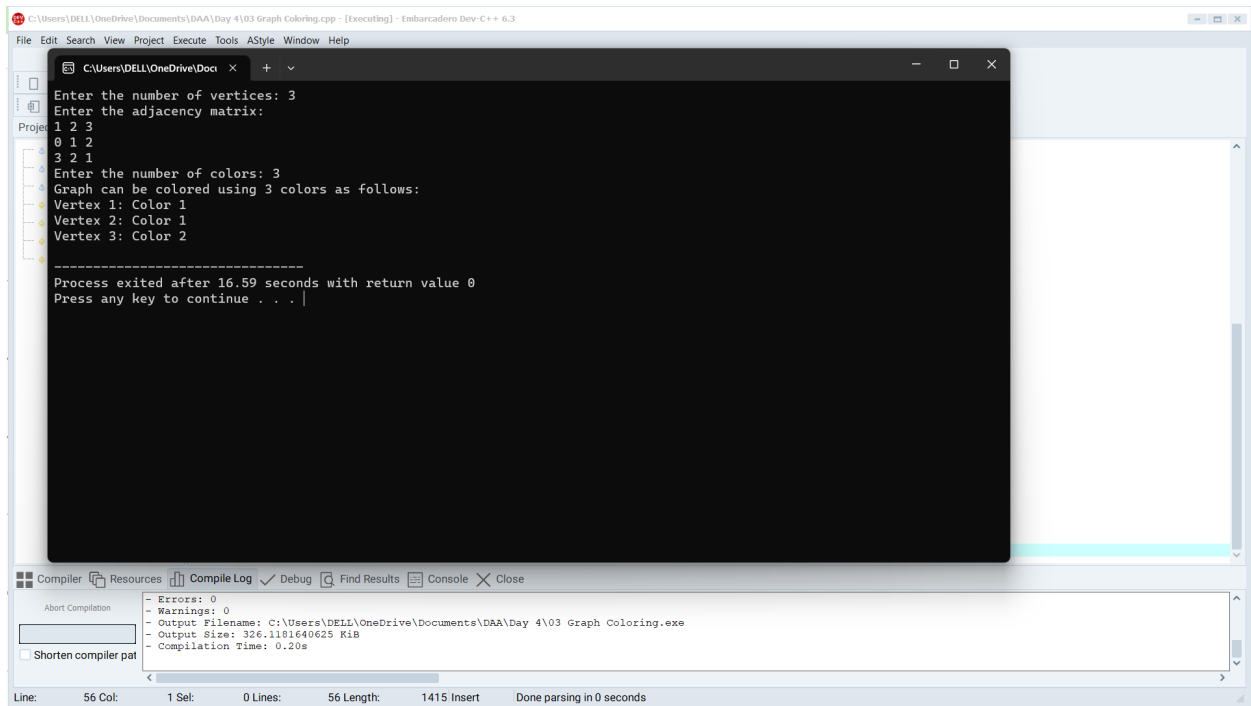
    scanf("%d", &numVertices);

    printf("Enter the adjacency matrix:\n");

```

```
    for (int i = 0; i < numVertices; i++) {  
        for (int j = 0; j < numVertices; j++) {  
            scanf("%d", &graph[i][j]);  
        }  
    }  
  
    printf("Enter the number of colors: ");  
    scanf("%d", &numColors);  
  
    if (graphColoring(0)) {  
        printf("Graph can be colored using %d colors as follows:\n", numColors);  
        for (int i = 0; i < numVertices; i++) {  
            printf("Vertex %d: Color %d\n", i + 1, colors[i]);  
        }  
    } else {  
        printf("Graph cannot be colored with %d colors.\n", numColors);  
    }  
  
    return 0;  
}
```

OUTPUT:



```
C:\Users\DELL\OneDrive\Documents\DAA\Day 4\03 Graph Coloring.cpp - [Executing] - Embarcadero Dev-C++ 6.3
File Edit Search View Project Execute Tools AStyle Window Help
C:\Users\DELL\OneDrive\Doc...
Enter the number of vertices: 3
Enter the adjacency matrix:
1 2 3
0 1 2
3 2 1
Enter the number of colors: 3
Graph can be colored using 3 colors as follows:
Vertex 1: Color 1
Vertex 2: Color 1
Vertex 3: Color 2
-----
Process exited after 16.59 seconds with return value 0
Press any key to continue . . . |
Compiler Resources Compile Log Debug Find Results Console Close
Abort Compilation
Shorten compiler pat
Errors: 0
Warnings: 0
Output Filename: C:\Users\DELL\OneDrive\Documents\DAA\Day 4\03 Graph Coloring.exe
Output Size: 326.1101640625 KiB
Compilation Time: 0.20s
Line: 56 Col: 1 Sel: 0 Lines: 56 Length: 1415 Insert Done parsing in 0 seconds
```

4. Write a program to compute container loader Problem.

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

```
#include <stdbool.h>
```

```
#define MAX_CONTAINERS 100
```

```
#define MAX_ITEMS 100
```

```
int containers[MAX_CONTAINERS];
```

```
int items[MAX_ITEMS];
```

```
int numContainers, numItems;
```

```
void containerLoading() {
```

```
    for (int i = 0; i < numItems - 1; i++) {
```

```
        for (int j = i + 1; j < numItems; j++) {
```

```
            if (items[i] < items[j]) {
```

```
                int temp = items[i];
```



```

        items[i] = items[j];

        items[j] = temp;
    }

}

}

int containerIndex = 0;

for (int i = 0; i < numContainers; i++) {

    containers[i] = 0;

}

for (int i = 0; i < numItems; i++) {

    bool placed = false;

    for (int j = 0; j <= containerIndex; j++) {

        if (containers[j] + items[i] <= 100) {

            containers[j] += items[i];

            placed = true;

            break;

        }

    }

    if (!placed) {

        containerIndex++;

        containers[containerIndex] = items[i];

    }

}

```

```
    printf("Container Loading Result:\n");  
    for (int i = 0; i <= containerIndex; i++) {  
        printf("Container %d: %d/%d\n", i + 1, containers[i], 100);  
    }  
}
```

```
int main() {  
    printf("Enter the number of containers: ");  
    scanf("%d", &numContainers);  
  
    printf("Enter the number of items: ");  
    scanf("%d", &numItems);  
  
    printf("Enter the sizes of items:\n");  
    for (int i = 0; i < numItems; i++) {  
        scanf("%d", &items[i]);  
    }  
  
    containerLoading();  
  
    return 0;  
}
```

OUTPUT:

```
Enter the number of containers: 2
Enter the number of items: 10
Enter the sizes of items:
100
250
3610
2541
320
156
312
250
20
100
Container Loading Result:
Container 1: 100/100
Container 2: 3610/100
Container 3: 2541/100
Container 4: 320/100
Container 5: 312/100
Container 6: 250/100
Container 7: 250/100
Container 8: 156/100
Container 9: 100/100
Container 10: 20/100

-----
Process exited after 20.3 seconds with return value 0
Press any key to continue . . .
```

5. Write a program to generate the list of all factor for n value.

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

```
int main() {
```

```
    int n;
```

```
    printf("Enter a positive integer: ");
```

```
    scanf("%d", &n);
```

```
    if (n <= 0) {
```

```
        printf("Please enter a positive integer.\n");
```

```
    } else {
```

```
        printf("Factors of %d are: ", n);
```

```
        for (int i = 1; i <= n; i++) {
```

```
            if (n % i == 0) {
```

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

```

    }

    }

    printf("\n");

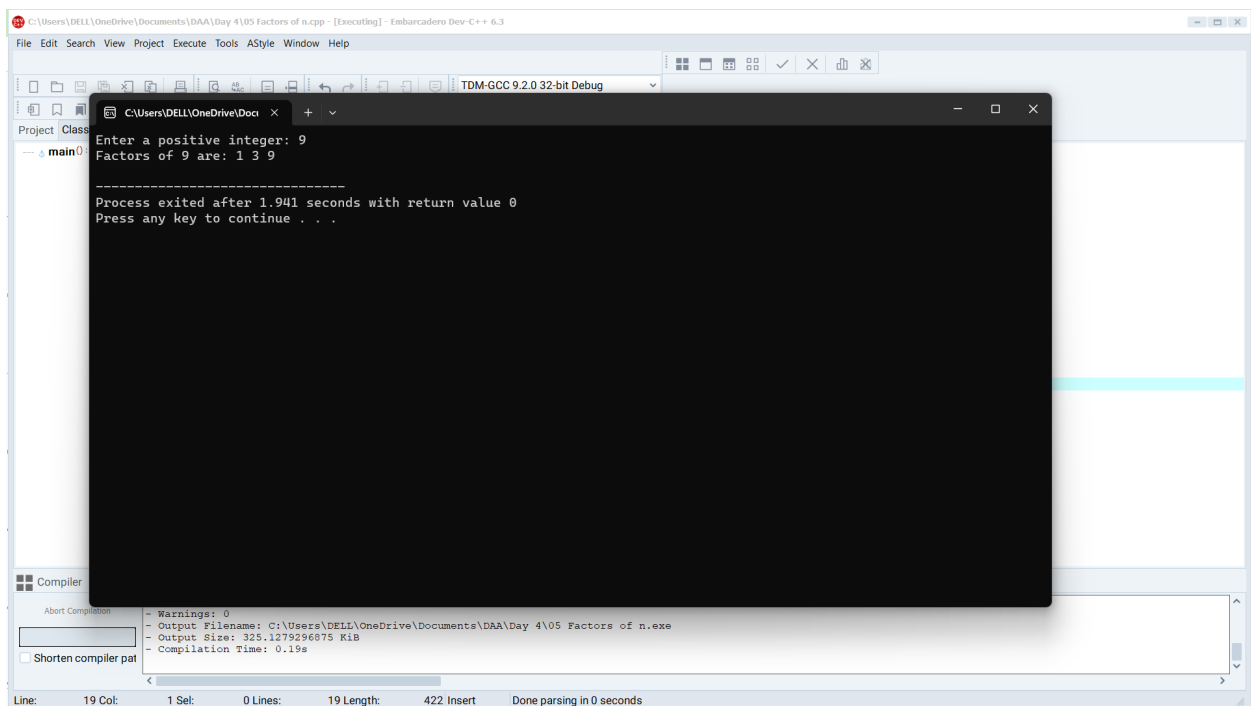
    }

    return 0;

}

```

OUTPUT:



6. Write a program to perform Assignment problem using branch and bound.

```

#include <stdio.h>

#include <limits.h>

#define N 5

int costMatrix[N][N];

int assignment[N];

```

```
int usedRows[N], usedCols[N];
```

```
int minCost = INT_MAX;
```

```
void printAssignment() {
```

```
    printf("Assignment:\n");
```

```
    for (int i = 0; i < N; i++) {
```

```
        printf("Agent %d is assigned to Task %d (Cost %d)\n", i + 1, assignment[i] + 1, costMatrix[i][assignment[i]]);
```

```
    }
```

```
}
```

```
void branchAndBound(int agent, int costSoFar) {
```

```
    if (agent == N) {
```

```
        if (costSoFar < minCost) {
```

```
            minCost = costSoFar;
```

```
            printAssignment();
```

```
        }
```

```
        return;
```

```
    }
```

```
    for (int task = 0; task < N; task++) {
```

```
        if (!usedCols[task] && (costSoFar + costMatrix[agent][task] < minCost)) {
```

```
            assignment[agent] = task;
```

```
            usedCols[task] = 1;
```

```
            branchAndBound(agent + 1, costSoFar + costMatrix[agent][task]);
```

```
        usedCols[task] = 0;
    }
}
}
```

```
int main() {
    printf("Enter the cost matrix (%d x %d):\n", N, N);
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            scanf("%d", &costMatrix[i][j]);
        }
    }

    for (int i = 0; i < N; i++) {
        assignment[i] = -1;
    }

    branchAndBound(0, 0);

    printf("Minimum Cost: %d\n", minCost);

    return 0;
}
```

OUTPUT:

```
C:\Users\DELL\OneDrive\Docu x + v
Agent 5 is assigned to Task 3 (Cost 1)
Assignment:
Agent 1 is assigned to Task 1 (Cost 1)
Agent 2 is assigned to Task 4 (Cost 6)
Agent 3 is assigned to Task 2 (Cost 0)
Agent 4 is assigned to Task 5 (Cost 6)
Agent 5 is assigned to Task 3 (Cost 1)
Assignment:
Agent 1 is assigned to Task 1 (Cost 1)
Agent 2 is assigned to Task 4 (Cost 6)
Agent 3 is assigned to Task 3 (Cost 2)
Agent 4 is assigned to Task 2 (Cost 0)
Agent 5 is assigned to Task 5 (Cost 4)
Assignment:
Agent 1 is assigned to Task 1 (Cost 1)
Agent 2 is assigned to Task 4 (Cost 6)
Agent 3 is assigned to Task 5 (Cost 3)
Agent 4 is assigned to Task 2 (Cost 0)
Agent 5 is assigned to Task 3 (Cost 1)
Assignment:
Agent 1 is assigned to Task 1 (Cost 1)
Agent 2 is assigned to Task 5 (Cost 2)
Agent 3 is assigned to Task 4 (Cost 4)
Agent 4 is assigned to Task 2 (Cost 0)
Agent 5 is assigned to Task 3 (Cost 1)
Minimum Cost: 8

-----
Process exited after 22.75 seconds with return value 0
Press any key to continue . . .
```

7. Write a program for to perform liner search.

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

```
int linearSearch(int arr[], int size, int target) {
```

```
    for (int i = 0; i < size; i++) {
```

```
        if (arr[i] == target) {
```

```
            return i;
```

```
        }
```

```
    }
```

```
    return -1;
```

```
}
```

```
int main() {
```

```
    int arr[100];
```

```
int size, target;

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

scanf("%d", &size);

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

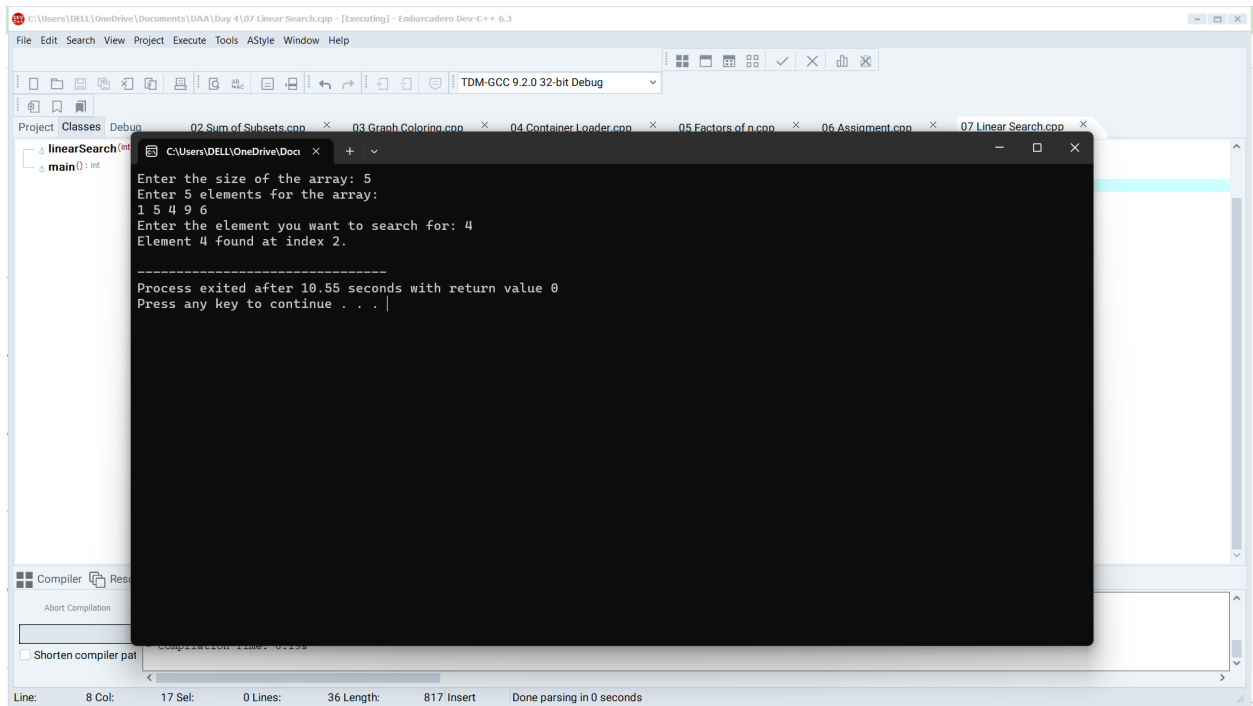
printf("Enter the element you want to search for: ");
scanf("%d", &target);

int result = linearSearch(arr, size, target);

if (result != -1) {
    printf("Element %d found at index %d.\n", target, result);
} else {
    printf("Element %d not found in the array.\n", target);
}

return 0;
}
```


OUTPUT:



```
C:\Users\DELL\OneDrive\Documents\DAAD\Day 4\07 Linear Search.cpp - [Executing] - Embarcadero Dev-C++ 6.3
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 9.2.0 32-bit Debug
Project Classes Debug 02 Sum of Subsets.cpp 03 Graph Coloring.cpp 04 Container Loader.cpp 05 Factors of n.cpp 06 Assignment.cpp 07 Linear Search.cpp
linearSearch.cpp
main0: int
Enter the size of the array: 5
Enter 5 elements for the array:
1 5 4 9 6
Enter the element you want to search for: 4
Element 4 found at index 2.

-----
Process exited after 10.55 seconds with return value 0
Press any key to continue . . .
```

8. Write a program to find out Hamiltonian circuit Using backtracking method

```
#include <stdio.h>

#include <stdbool.h>

#define V 5

int path[V];

bool visited[V];

void printHamiltonianCircuit() {

    printf("Hamiltonian Circuit: ");

    for (int i = 0; i < V; i++) {

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

    }

    printf("%d\n", path[0]);
```

```
}
```

```
bool isSafe(int v, int pos, int graph[V][V]) {
```

```
    if (!graph[path[pos - 1]][v])
```

```
        return false;
```

```
    for (int i = 0; i < pos; i++) {
```

```
        if (path[i] == v)
```

```
            return false;
```

```
    }
```

```
    return true;
```

```
}
```

```
bool hamiltonianCircuitUtil(int graph[V][V], int pos) {
```

```
    if (pos == V) {
```

```
        if (graph[path[pos - 1]][path[0]] == 1) {
```

```
            printHamiltonianCircuit();
```

```
            return true;
```

```
        }
```

```
        return false;
```

```
    }
```

```
    for (int v = 1; v < V; v++) {
```

```
        if (!visited[v] && isSafe(v, pos, graph)) {
```

```
            path[pos] = v;
```

```

        visited[v] = true;

        if (hamiltonianCircuitUtil(graph, pos + 1))
            return true;

        path[pos] = -1;
        visited[v] = false;
    }
}

return false;
}

bool findHamiltonianCircuit(int graph[V][V]) {
    for (int i = 0; i < V; i++) {
        path[i] = -1;
        visited[i] = false;
    }

    path[0] = 0;
    visited[0] = true;

    if (hamiltonianCircuitUtil(graph, 1) == false) {
        printf("No Hamiltonian Circuit exists\n");
    }
}

```

```
        return false;
    }

    return true;
}

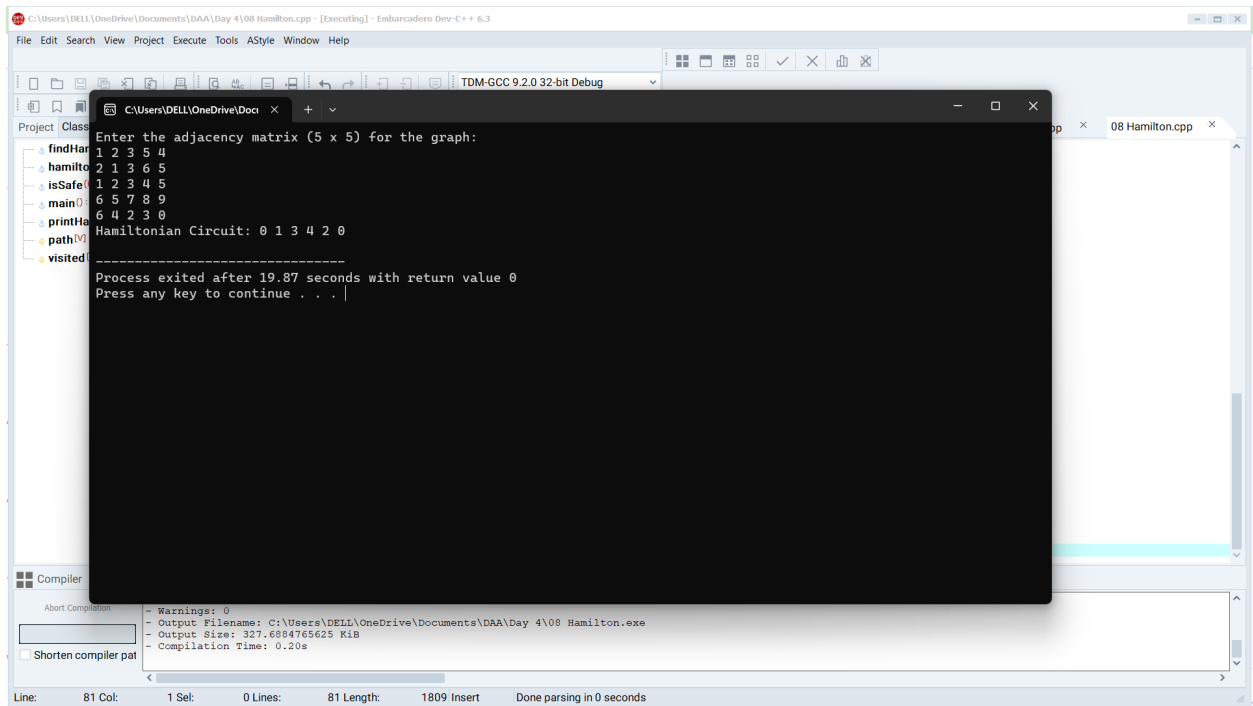
int main() {
    int graph[V][V];

    printf("Enter the adjacency matrix (%d x %d) for the graph:\n", V, V);
    for (int i = 0; i < V; i++) {
        for (int j = 0; j < V; j++) {
            scanf("%d", &graph[i][j]);
        }
    }

    if (findHamiltonianCircuit(graph) == false)
        printf("No Hamiltonian Circuit exists\n");

    return 0;
}
```

OUTPUT:



```
C:\Users\DELL\OneDrive\Documents\DAA\Day 4\08 Hamilton.cpp - [Executing] - Embarcadero Dev-C++ 6.3
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 9.2.0 32-bit Debug
Project Class
  findHar
  hamilton
  isSafe
  main
  printHa
  path
  visited
Enter the adjacency matrix (5 x 5) for the graph:
1 2 3 5 4
2 1 3 6 5
1 2 3 4 5
6 5 7 8 9
6 4 2 3 0
Hamiltonian Circuit: 0 1 3 4 2 0
-----
Process exited after 19.87 seconds with return value 0
Press any key to continue . . . |
Compiler
  Warnings: 0
  Output Filename: C:\Users\DELL\OneDrive\Documents\DAA\Day 4\08 Hamilton.exe
  Output Size: 327.6884765625 KiB
  Compilation Time: 0.20s
Shorten compiler path
Line: 81 Col: 1 Sel: 0 Lines: 81 Length: 1809 Insert Done parsing in 0 seconds
```