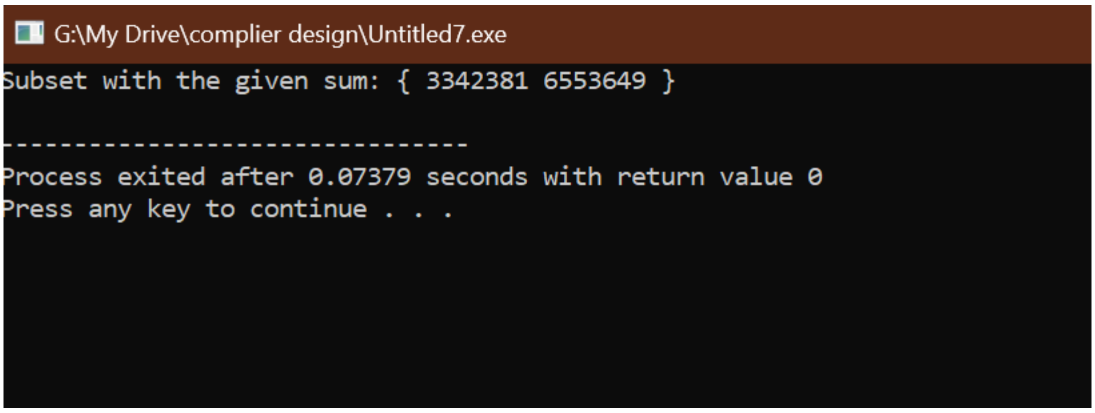
36, Write a program to perform graph coloring problem using backtracking.

Code:

```
#include <stdio.h>
#include <stdbool.h>
#define V 4
void printSolution(int color[]) {
    printf("Vertex colors: ");
    for (int i = 0; i < V; i++) {
        printf("%d ", color[i]);
    }
    printf("\n");
}
bool isSafe(int graph[V][V], int v, int color[], int c) {
    for (int i = 0; i < V; i++) {
        if (graph[v][i] && c == color[i]) {
            return false;
        }
    }
    return true;
}
bool graphColoringUtil(int graph[V][V], int m, int color[], int v) {
    if (v == V) {
        printSolution(color);
        return true;
    }
    for (int c = 1; c <= m; c++) {
        if (isSafe(graph, v, color, c)) {
            color[v] = c;

            if (graphColoringUtil(graph, m, color, v + 1)) {
                return true;
            }

            color[v] = 0;
        }
    }
    return false;
}
void graphColoring(int graph[V][V], int m) {
    int color[V];
    for (int i = 0; i < V; i++) {
        color[i] = 0;
    }
    if (!graphColoringUtil(graph, m, color, 0)) {
        printf("Solution does not exist.\n");
    }
}
int main() {
    int graph[V][V] = {
        {0, 1, 1, 1},
        {1, 0, 1, 0},
        {1, 1, 0, 1},
        {1, 0, 1, 0}
    };
    int m = 3;
    graphColoring(graph, m);
    return 0;
}
```

Output:



G:\My Drive\complier design\Untitled7.exe

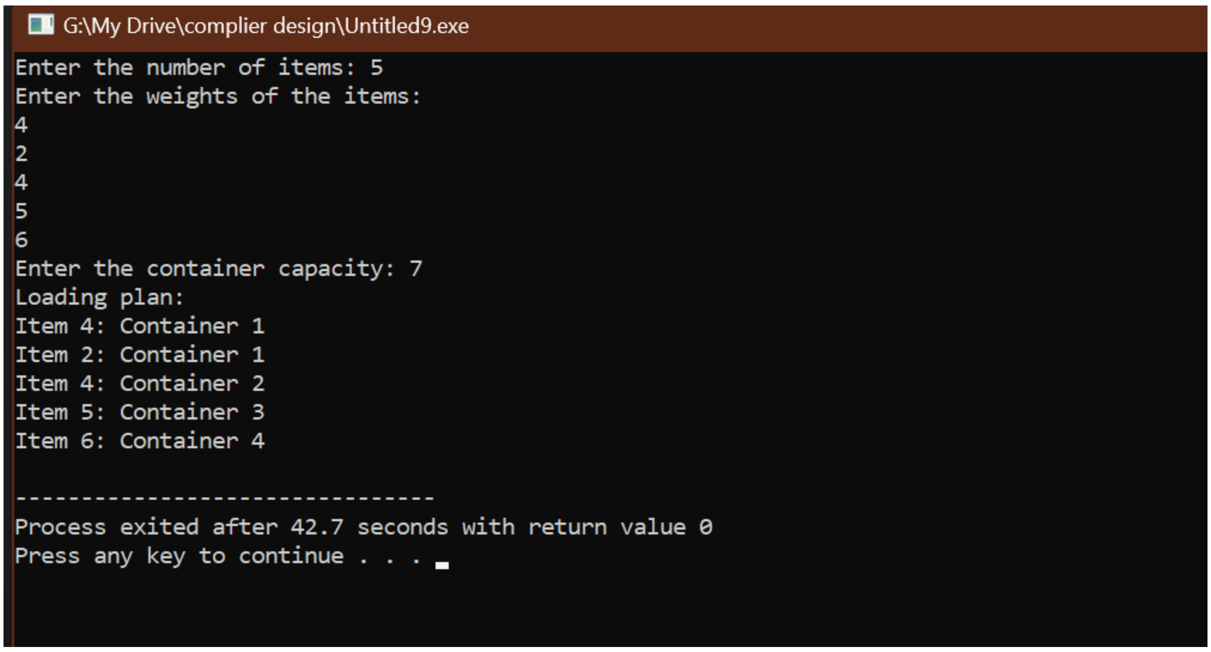
```
Subset with the given sum: { 3342381 6553649 }

--------------------------------
Process exited after 0.07379 seconds with return value 0
Press any key to continue . . .
```

37, Write a program to compute container loader Problem.

Code:

```
#include <stdio.h>
#define MAX_ITEMS 100
void containerLoader(int items[], int numItems, int containerCapacity) {
    int currentContainer = 1;
    int currentWeight = 0;
    printf("Loading plan:\n");
    for (int i = 0; i < numItems; i++) {
        if (currentWeight + items[i] <= containerCapacity) {
            printf("Item %d: Container %d\n", items[i], currentContainer);
            currentWeight += items[i];
        } else {
            currentContainer++;
            currentWeight = items[i];
            printf("Item %d: Container %d\n", items[i], currentContainer);
        }
    }
}
int main() {
    int items[MAX_ITEMS];
    int numItems, containerCapacity;
    printf("Enter the number of items: ");
    scanf("%d", &numItems);
    printf("Enter the weights of the items:\n");
    for (int i = 0; i < numItems; i++) {
        scanf("%d", &items[i]);
    }
    printf("Enter the container capacity: ");
    scanf("%d", &containerCapacity);
    containerLoader(items, numItems, containerCapacity);
    return 0;
}
```

Output:



G:\My Drive\complier design\Untitled9.exe

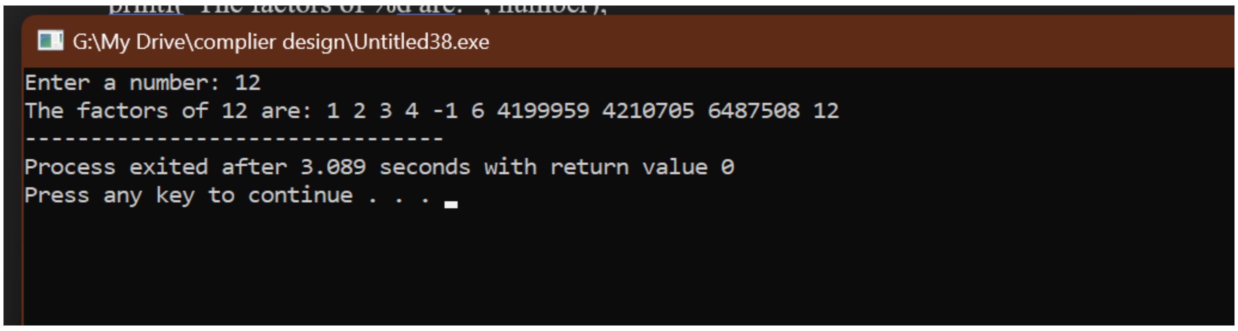
```
Enter the number of items: 5
Enter the weights of the items:
4
2
4
5
6
Enter the container capacity: 7
Loading plan:
Item 4: Container 1
Item 2: Container 1
Item 4: Container 2
Item 5: Container 3
Item 6: Container 4

--------------------------------
Process exited after 42.7 seconds with return value 0
Press any key to continue . . .
```

38, Write a program to generate the list of all factor for n value using recursion

Code:

```
#include <stdio.h>
void findFactors(int n, int currentFactor, int factors[]) {
    if (currentFactor > n) {
        return;
    }

    if (n % currentFactor == 0) {
        factors[currentFactor - 1] = currentFactor;
    }

    findFactors(n, currentFactor + 1, factors);
}
int main() {
    int number;

    printf("Enter a number: ");
    scanf("%d", &number);

    int factors[number];
    findFactors(number, 1, factors);

    printf("The factors of %d are: ", number);
    for (int i = 0; i < number; i++) {
        if (factors[i] != 0) {
            printf("%d ", factors[i]);
        }
    }

    return 0;
}
```

Output:



```
Enter a number: 12
The factors of 12 are: 1 2 3 4 -1 6 4199959 4210705 6487508 12
--------------------------------
Process exited after 3.089 seconds with return value 0
Press any key to continue . . .
```

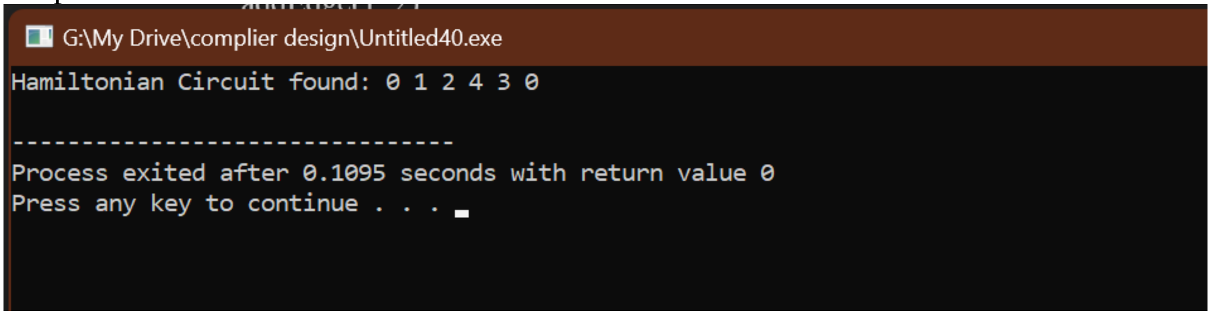
40 , Write a program to find out Hamiltonian circuit Using backtracking method.

Code:

```
#include <stdio.h>

#define MAX_VERTICES 10

int numVertices;
int graph[MAX_VERTICES][MAX_VERTICES];
int path[MAX_VERTICES];
```

```
void initializeGraph() {
    int i, j;
    for (i = 0; i < MAX_VERTICES; i++) {
        for (j = 0; j < MAX_VERTICES; j++) {
            graph[i][j] = 0;
        }
    }
}

void addEdge(int from, int to) {
    graph[from][to] = 1;
    graph[to][from] = 1;
}

void printSolution() {
    int i;
    printf("Hamiltonian Circuit found: ");
    for (i = 0; i < numVertices; i++) {
        printf("%d ", path[i]);
    }
    printf("%d\n", path[0]);  // Print the starting vertex to complete the circuit
}

int isSafe(int v, int pos) {
    if (graph[path[pos - 1]][v] == 0)
        return 0;

    for (int i = 0; i < pos; i++)
        if (path[i] == v)
            return 0;

    return 1;
}

int hamiltonianUtil(int pos) {
    if (pos == numVertices) {
        if (graph[path[pos - 1]][path[0]] == 1)
            return 1;
        else
            return 0;
    }

    for (int v = 1; v < numVertices; v++) {
        if (isSafe(v, pos)) {
            path[pos] = v;
            if (hamiltonianUtil(pos + 1))
                return 1;
            path[pos] = -1;
        }
    }

    return 0;
}

void findHamiltonianCircuit() {
    int i;
    for (i = 0; i < MAX_VERTICES; i++) {
        path[i] = -1;
    }

    path[0] = 0;

    if (hamiltonianUtil(1)) {
        printSolution();
    } else {
        printf("No Hamiltonian Circuit exists.\n");
    }
}

int main() {
    numVertices = 5;
    initializeGraph();

    addEdge(0, 1);
    addEdge(0, 3);
    addEdge(1, 2);
    addEdge(1, 3);
    addEdge(2, 4);
    addEdge(3, 4);

    findHamiltonianCircuit();

    return 0;
}
```

Output:



```
Hamiltonian Circuit found: 0 1 2 4 3 0

--------------------------------
Process exited after 0.1095 seconds with return value 0
Press any key to continue . . .
```