Program 8

Design and implement C Program to find a subset of a given set S = {sl , s2,. ...,sn} of n positive integers whose sum is equal to a given positive integer d.

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

#define MAX_SIZE 100

// Function to find subset with given sum
void subsetSum(int set[], int subset[], int n, int subSize, int total, int nodeCount, int sum) {
    if (total == sum) {
        // Print the subset
        printf("Subset found: { ");
        for (int i = 0; i < subSize; i++) {
            printf("%d ", subset[i]);
        }
        printf("}\n");
        return;
    } else {
        // Check the sum of the remaining elements
        for (int i = nodeCount; i < n; i++) {
            subset[subSize] = set[i];
            subsetSum(set, subset, n, subSize + 1, total + set[i], i + 1, sum);
        }
    }
}
int main() {
    int set[MAX_SIZE];
    int subset[MAX_SIZE];
    int n, sum;
```

```
    // Input the number of elements in the set
    printf("Enter the number of elements in the set: ");
    scanf("%d", &n);

    // Input the elements of the set
    printf("Enter the elements of the set:\n");
    for (int i = 0; i < n; i++) {
        scanf("%d", &set[i]);
    }

    // Input the target sum
    printf("Enter the sum to find subset for: ");
    scanf("%d", &sum);

    printf("Subsets with sum %d:\n", sum);
    subsetSum(set, subset, n, 0, 0, 0, sum);

    return 0;
}
```

OUTPUT:

```
student@lenovo-ThinkCentre-M900:~$ gcc program8.c
student@lenovo-ThinkCentre-M900:~$ ./a.out
Enter the number of elements in the set: 5
Enter the elements of the set:
2
4
6
8
10
Enter the sum to find subset for: 10
Subsets with sum 10:
Subset found: { 2 8 }
Subset found: { 4 6 }
Subset found: { 10 }
```

Program 5

Design and implement C Program to obtain the Topological ordering of vertices in a given digraph.

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

#define MAX_VERTICES 100

// Structure to represent a graph
typedef struct {
    int V;
    int** adjMatrix;
} Graph;

// Function to create a new graph
Graph* createGraph(int V) {
    Graph* graph = (Graph*)malloc(sizeof(Graph));
    graph->V = V;
    graph->adjMatrix = (int**)calloc(V, sizeof(int*));
    for (int i = 0; i < V; i++) graph->adjMatrix[i] = (int*)calloc(V, sizeof(int));
    return graph;
}

// Function to add an edge to the graph
void addEdge(Graph* graph, int src, int dest) {
    graph->adjMatrix[src][dest] = 1;
}

// Function to perform topological sorting
void topologicalSort(Graph* graph) {
    int V = graph->V, inDegree[MAX_VERTICES] = {0}, queue[MAX_VERTICES], front = 0, rear =
-1;
```

```
    for (int i = 0; i < V; i++)
        for (int j = 0; j < V; j++)
            if (graph->adjMatrix[i][j] == 1) inDegree[j]++;

    for (int i = 0; i < V; i++) if (inDegree[i] == 0) queue[++rear] = i;

    printf("Topological ordering of vertices: ");
    while (front <= rear) {
        int vertex = queue[front++];
        printf("%d ", vertex);
        for (int i = 0; i < V; i++) if (graph->adjMatrix[vertex][i] == 1 && --inDegree[i] == 0)
queue[++rear] = i;
    }
    printf("\n");
}

// Driver code
int main() {
    int V, E;
    printf("Enter the number of vertices: ");
    scanf("%d", &V);
    Graph* graph = createGraph(V);
    printf("Enter the number of edges: ");
    scanf("%d", &E);
    printf("Enter the edges (source vertex, destination vertex):\n");
    for (int i = 0, src, dest; i < E; i++) {
        scanf("%d %d", &src, &dest);
        addEdge(graph, src, dest);
    }
    topologicalSort(graph);
    return 0;
}
```