CSA0672-DAA-DAY3

K.Dinesh

192011131

1.Write a program to return all the possible subsets for a given integer array. Return the solution in any order.

Input nums= [1,2,3]

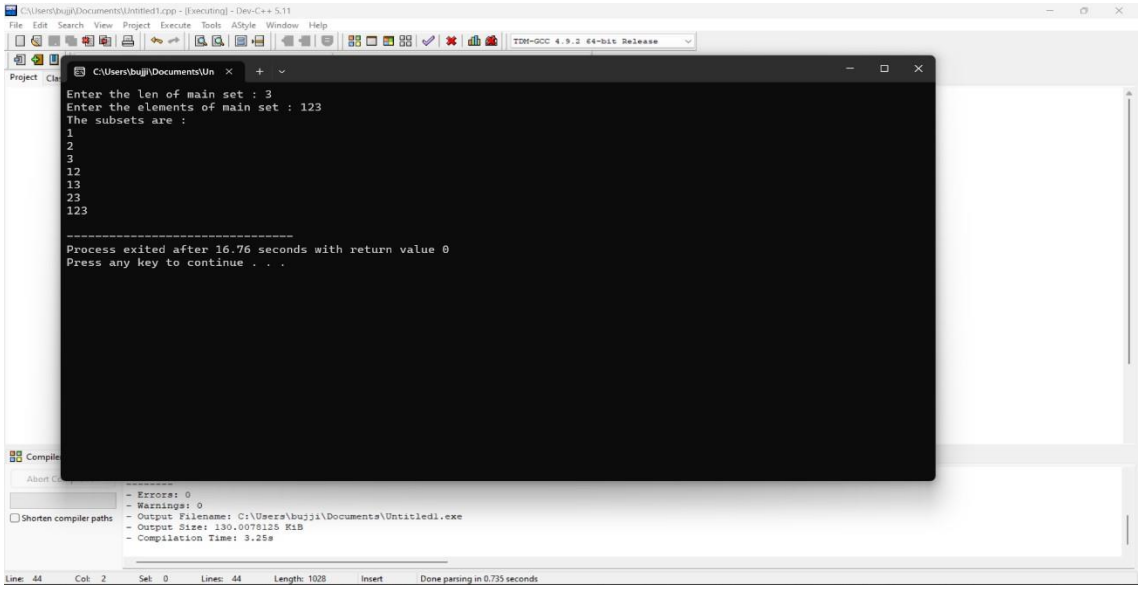
Output : [[], [1], [2], [3], [1,2], [1,3], [2,3], [1,2,3]] Program

:

```
#include <stdio.h> char
string[50], n; void
subset(int, int, int);  int
main()
{    int i,
len;
   printf("Enter the len of main set : ");
scanf("%d", &len);    printf("Enter the
elements of main set : ");    scanf("%s",
string);    n = len;
   printf("The subsets are :\n");
   for (i = 1;i <= n;i++)
subset(0, 0, i);
}
void subset(int start, int index, int num_sub)
{    int i,
j;
   if (index - start + 1  ==  num_sub)
   {
```

```
      if (num_sub  ==  1)
      {
         for     (i     =     0;i     <     n;i++)
printf("%c\n", string[i]);        }
      else
      {
         for (j = index;j < n;j++)
         {
            for (i = start;i < index;i++)
printf("%c", string[i]);                printf("%c\n",
string[j]);
         }
         if (start != n - num_sub)
subset(start + 1, start + 1, num_sub);
      }
   }
   else
   {
      subset(start, index + 1, num_sub);
   }
}
```

Output :



2.Write a program to perform sum of subsets problem using backtracking and estimate time complexity. Identify the test cases.

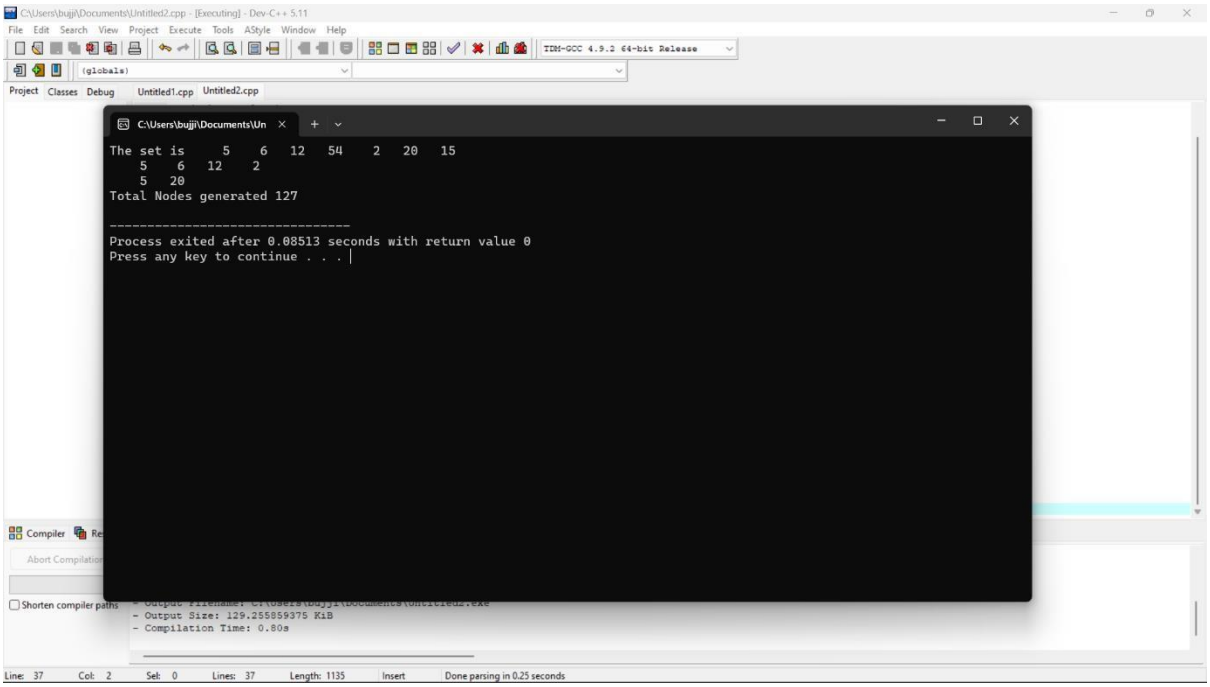
A. Set (s) = (6, 2,8,1,5) sum is 9 B. Set (s) = (6, -4, 7,-1,5, 2,8,1,) sum is 10

Program :

```
#include <stdio.h> #include
<stdlib.h> static int total_nodes;
void printValues(int A[], int
size){
  for (int i = 0; i < size; i++) {
printf("%*d", 5, A[i]);
  }
  printf("\n");
}
void subset_sum(int s[], int t[], int s_size, int t_size, int sum, int ite, int const
target_sum){   total_nodes++;   if (target_sum == sum) {      printValues(t, t_size);
subset_sum(s, t, s_size, t_size - 1, sum - s[ite], ite + 1, target_sum);
    return;
  }
  else {      for (int i = ite; i <
s_size; i++) {         t[t_size] = s[i];
```

```
        subset_sum(s, t, s_size, t_size + 1, sum + s[i], i + 1, target_sum);
      }
    }
}
void generateSubsets(int s[], int size, int target_sum){
int* tuplet_vector = (int*)malloc(size * sizeof(int));
subset_sum(s, tuplet_vector, size, 0, 0, 0, target_sum);
free(tuplet_vector);
}
int main(){
  int set[] = { 5, 6, 12 , 54, 2 , 20 , 15 };    int size =
sizeof(set) / sizeof(set[0]);    printf("The set is ");
printValues(set , size);    generateSubsets(set, size,
25);    printf("Total Nodes generated %d\n",
total_nodes);
  return 0;
}
```

Output :



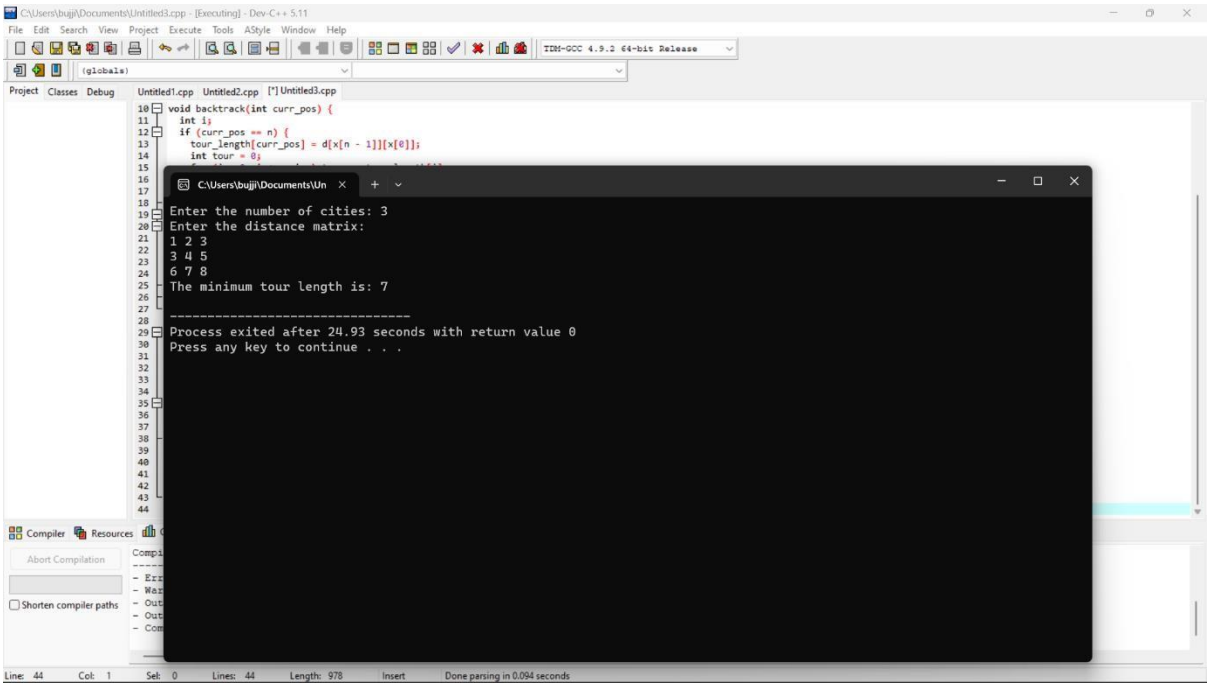
3.Determine an optimal tour in a weighted, directed graph. The weights are nonnegative numbers. The inputs are weighted, directed graph, and n, the number of vertices in the graph. The graph is represented by a twodimensional array W, which has both its rows and columns indexed from 1 to n, where W [i] [j] is the weight on the edge from the ith vertex to the jth vertex. Write a program for travelling salesman problem using dynamic programming for the below given graph.

Program :

```
#include <stdio.h>
#include <stdbool.h>
#define MAX 20 #define INF 99999 int n,
d[MAX][MAX], x[MAX]; int best_tour_length
= INF, tour_length[MAX]; void backtrack(int
curr_pos) {
 int i;
 if (curr_pos == n) {
tour_length[curr_pos] = d[x[n - 1]][x[0]];
   int tour = 0;    for (i = 0; i < n; i++) tour +=
tour_length[i];    if (tour < best_tour_length)
best_tour_length = tour;
   return;
 }
 for (i = 0; i < n; i++) {    if (x[i] == -1) {      x[i] =
curr_pos;      tour_length[curr_pos] =
d[x[curr_pos - 1]][i];      backtrack(curr_pos +
1);
     x[i] = -1;
   }
```

```
  }
}
int main() {
  int i, j;
  printf("Enter the number of cities: ");
scanf("%d", &n);   printf("Enter the
distance matrix:\n");
  for (i = 0; i < n; i++)
for (j = 0; j < n; j++) {
scanf("%d", &d[i][j]);
      x[i] = -1;
    }
  x[0] = 0;   backtrack(1);   printf("The minimum tour length is:
%d\n", best_tour_length);
  return 0;
}
```

Output :



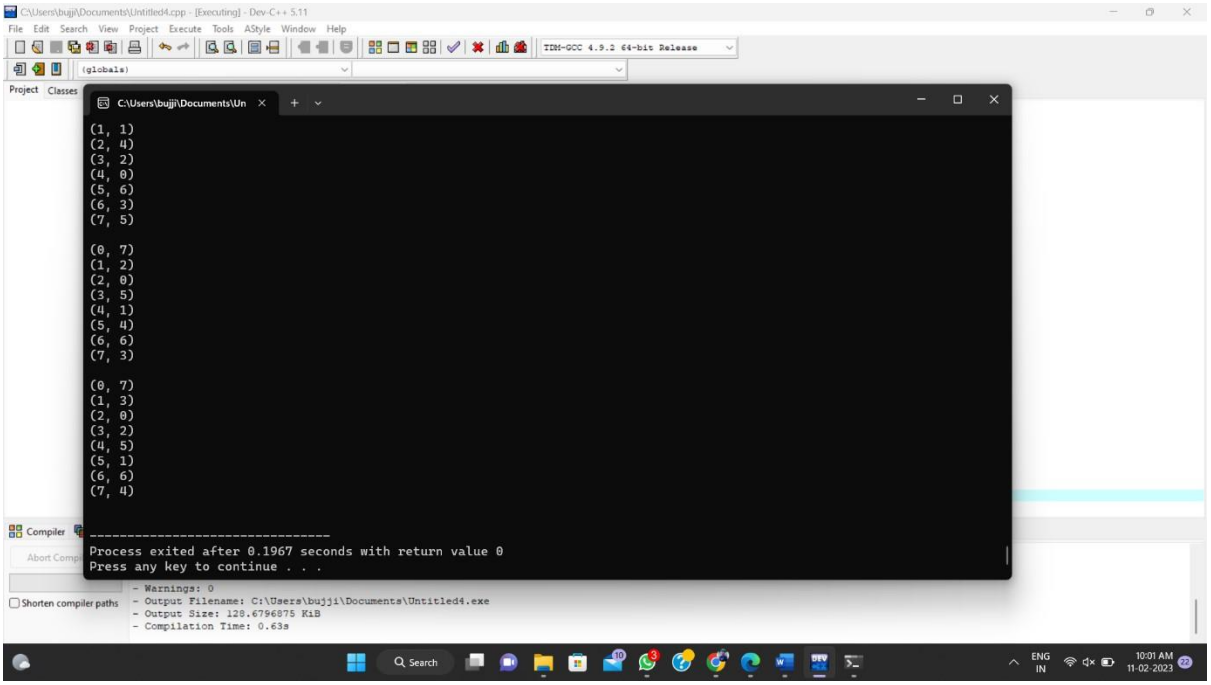
4.The n-queens puzzle is the problem of placing n queens on an n x n chessboard such that no two queens attack each other. Given an integer n, return all distinct solutions to the n-queens puzzle. You may return the answer in any order. Write a program for the same.

Program :

```
#include <stdio.h>
#include <stdbool.h> #define
N 8
int col[N]; bool
check(int row) {
  int i;
  for (i = 0; i < row; i++)     if
(col[i] == col[row] ||         row - i
== col[row] - col[i] ||         row -
i == col[i] - col[row])       return
false;   return true;
}
void backtrack(int row) {
  int i;
  if (row == N) {     for (i = 0; i < N; i++) printf("(%d,
%d)\n", i, col[i]);     printf("\n");     return;
  }
  for (i = 0; i < N; i++) {
col[row] = i;
    if (check(row)) backtrack(row + 1);
  }
}
int main() {
backtrack(0);
return 0;
```

}

Output :



5.Write a program to perform Minimum spanning tree using greedy techniques and estimate time complexity for the given set of values.

Program :

```
#include <stdio.h>
#include <limits.h>  #define V 5
int minKey(int key[], int mstSet[])
{    int min = INT_MAX,
min_index;
   int v;
   for (v = 0; v < V; v++)
      if (mstSet[v] == 0 && key[v] < min)
min = key[v], min_index = v;     return
min_index;
```

```
}
int printMST(int parent[], int n, int graph[V][V]) {
   int i;
   printf("Edge   Weight\n");
   for (i = 1; i < V; i++)        printf("%d - %d    %d \n", parent[i],
i, graph[i][parent[i]]);
}
void primMST(int graph[V][V]) {
   int parent[V];     int
key[V], i, v, count;
int mstSet[V];  for (v =
0; v < V; v++)

          if (graph[u][v] && mstSet[v] == 0 && graph[u][v] < key[v])
parent[v] = u, key[v] = graph[u][v];
   }
   printMST(parent, V, graph);
}
int main() {
    2    3
   (0)--(1)--(2)
|   / \   |
   6| 8/   \5 |7
    | /     \ |
   (3)-------(4)
    9         */
   int graph[V][V] = { { 0, 2, 0, 6, 0 }, { 2, 0, 3, 8, 5 },
{ 0, 3, 0, 0, 7 }, { 6, 8, 0, 0, 9 }, { 0, 5, 7, 9, 0 }, };
   primMST(graph);
```

```
    return 0;
}
```

Output :