1. Identify the M-th maximum number and Nth minimum number in an array and then find the sum of it and difference of it.

Test cases:

output –

a. {16, 16, 16 16, 16}, M = 0, N = 1 (illegal input)

b. {0, 0, 0, 0}, M = 1, N = 2 0

c. {-12, -78, -35, -42, -85}, M = 3 , N = 3 -7

d. {15, 19, 34, 56, 12}, M = 6 , N = -3 (illegal input)

e. {85, 45, 65, 75, 95}, M = 5 , N = 2              -20

program:

#include <stdio.h>

int main ()

{

int number[30],rev[30];

int i, j, a, n,m,nt;

printf("Enter the value of N\n");

scanf("%d", &n);

printf("Enter the numbers \n");

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

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

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

{

for (j = i + 1; j < n; ++j)

{

if (number[i] < number[j])

{

a = number[i];

number[i] = number[j];

number[j] = a;

}

}

}

for(i=(n-1),j=0; i>=0; i--,j++)

{

rev[j]=number[i];

}

printf("\nEnter the Mth maximum =");

scanf("%d",&m);

printf("\n%d th maximum number is %d",m,number[m-1]);

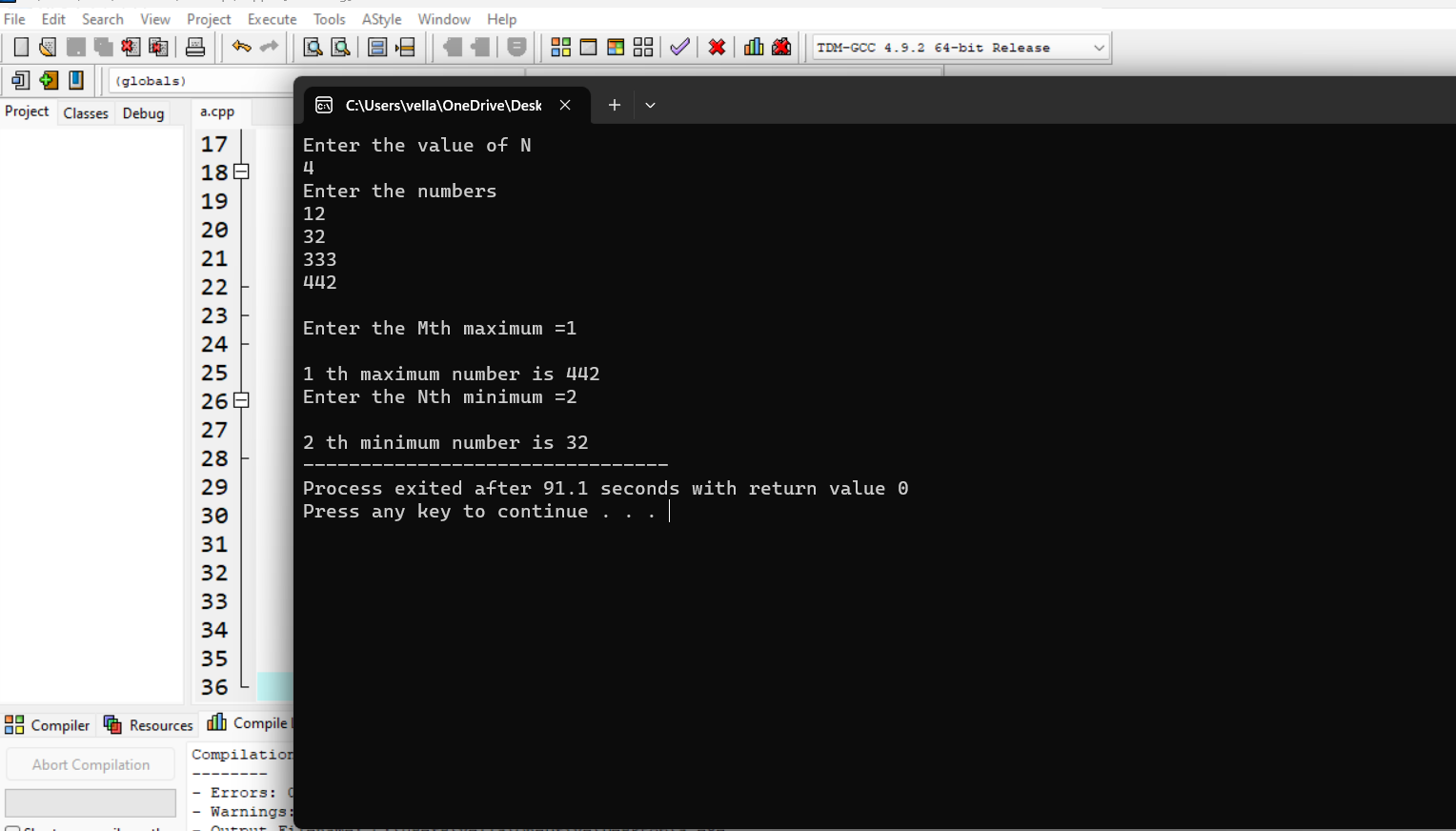
printf("\nEnter the Nth minimum =");

scanf("%d",&nt);

printf("\n%d th minimum number is %d",nt,rev[nt-1]);

return 0;

}



1. 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

#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 ");

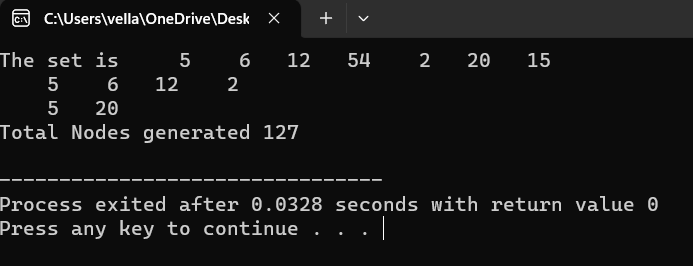
printValues(set , size);

generateSubsets(set, size, 25);

printf("Total Nodes generated %d\n", total\_nodes);

return 0;

}



3.) Write a program to find a minimum spanning tree using prims technique for the given graph Program:

#include <stdio.h>

#include <limits.h>

#define vertices 5

int minimum\_key(int k[], int mst[])

{

int minimum = INT\_MAX, min,i,count=0;

for (i = 0; i < vertices; i++)

if (mst[i] == 0 && k[i] < minimum )

minimum = k[i], min = i;

return min;

count++;

}

void prim(int g[vertices][vertices])

{

int parent[vertices];

int k[vertices];

int mst[vertices];

int i, count,edge,v;

for (i = 0; i < vertices; i++)

{

k[i] = INT\_MAX;

count++;

mst[i] = 0;

count++;

}

count++;

k[0] = 0;

count++;

parent[0] = -1;

count++;

for (count = 0; count < vertices-1; count++)

{

edge = minimum\_key(k, mst);

mst[edge] = 1;

for (v = 0; v < vertices; v++)

{

if (g[edge][v] && mst[v] == 0 && g[edge][v] < k[v])

{

parent[v] = edge, k[v] = g[edge][v];

}

}

}

count++;

count++;

count++;

printf("\n Edge \t Weight\n");

for (i = 1; i < vertices; i++)

printf(" %d <-> %d %d \n", parent[i], i, g[i][parent[i]]);

count++;

printf(" time complexity is :%d",count);

}

int main()

{

int i,j,g[vertices][vertices];

for (i=0;i<5;i++)

{

for(j=0;j<5;j++)

{

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

}

}

prim(g);

return 0;

}

4) Write a program to perform Knapsack problem for the following set of object values., Knapsack weight 100 item

Weight Profit

40 80

30 70

20 50

30 8

#include <stdio.h>

#include <stdlib.h>

#define MAX\_ITEMS 100

#define MAX\_WEIGHT 100

int weight[MAX\_ITEMS];

int value[MAX\_ITEMS];

int dp[MAX\_ITEMS][MAX\_WEIGHT];

int max(int a, int b) {

return (a > b) ? a : b;

}

int knapsack(int n, int w) {

int i, j;

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

for (j = 0; j <= w; j++) {

if (i == 0 || j == 0) {

dp[i][j] = 0;

} else if (weight[i-1] <= j) {

dp[i][j] = max(value[i-1] + dp[i-1][j-weight[i-1]], dp[i-1][j]);

} else {

dp[i][j] = dp[i-1][j];

}

}

}

return dp[n][w];

}

int main()

{

int n,w,i;

printf("Enter N :");

scanf("%d",&n);

printf("Enter weight :");

scanf("%d",&w);

printf("Enter Weights of %d bags :",n);

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

{

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

}

printf("Enter values of %d bags :",n);

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

{

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

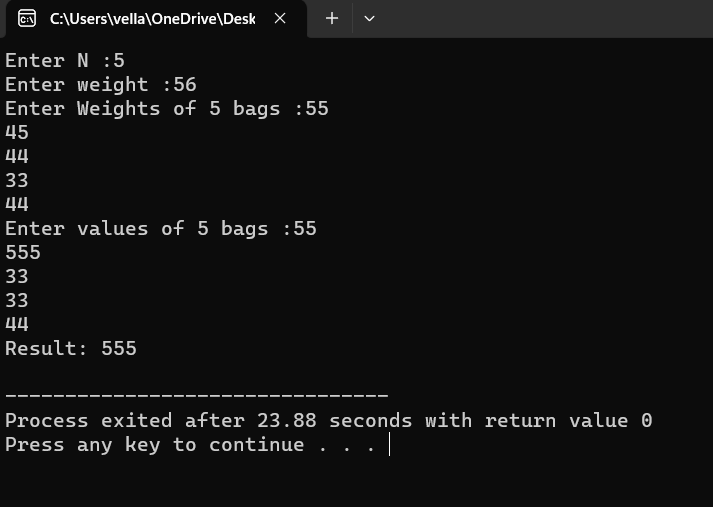
}

int result = knapsack(n, w);

printf("Result: %d\n", result);

return 0;

}



5) Write a program to compute container loader Problem for the given values and estimate time complexity. N=8 be total no of containers having weights (w1, w2, w3,…w8) = [ 50, 100, 30, 80, 90, 200, 150, 20 ]. Capacity value = 100

PROGRAM:

#include<stdio.h>

int main()

{

int c=0;

int n,e,w[20],w1[20],x[20],i,j,k,j1=0;

c++;

printf("Enter Strip Capacity : ");

scanf("%d",&e);

printf("Enter No of Containers : ");

scanf("%d",&n);

printf("Enter Containers weights : \n");

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

{

c++;

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

}

c++;

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

{

c++;

x[i]=0;

}

c++;

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

{

c++;

w1[i]=w[i];

}

c++;

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

{

c++;

for(j=0;j<n;j++)

{

c++;

c++;

if(w[i]<w[j])

{

k=w[i];

c++;

w[i]=w[j];

c++;

w[j]=k;

c++;

}

}

c++;

}

c++;

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

{

c++;

c++;

if(e>w[i])

{

e=e-w[i];

c++;

for(j=0;j<n;j++)

{

c++;

c++;

if(w[i]==w1[j])

{

x[j]=1;

c++;

}

}

c++;

}

}

c++;

printf("Container Loading :\n");

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

{

c++;

printf("%d\t",x[i]);

}

c++;

printf("\nTime Complexity : %d",c);

}

