**DATA STRUCTURE**

**MINI PROJECT**

**By**

**C. S. Roy Jasper**

**||-Year** B. Tech IT

**14bit058**

**Question 1:**

**FCTRL**

The most important part of a GSM network is so called *Base Transceiver Station* (*BTS*). These transceivers form the areas called *cells* (this term gave the name to the cellular phone) and every phone connects to the BTS with the strongest signal (in a little simplified view). Of course, BTSes need some attention and technicians need to check their function periodically.

ACM technicians faced a very interesting problem recently. Given a set of BTSes to visit, they needed to find the shortest path to visit all of the given points and return back to the central company building. Programmers have spent several months studying this problem but with no results. They were unable to find the solution fast enough. After a long time, one of the programmers found this problem in a conference article. Unfortunately, he found that the problem is so called "Travelling Salesman Problem" and it is very hard to solve. If we have *N* BTSes to be visited, we can visit them in any order, giving us *N*! possibilities to examine. The function expressing that number is called factorial and can be computed as a product 1.2.3.4....*N*. The number is very high even for a relatively small *N*.

The programmers understood they had no chance to solve the problem. But because they have already received the research grant from the government, they needed to continue with their studies and produce at least *some* results. So they started to study behaviour of the factorial function.

For example, they defined the function *Z*. For any positive integer *N*, *Z*(*N*) is the number of zeros at the end of the decimal form of number *N*!. They noticed that this function never decreases. If we have two numbers *N* 1 <*N* 2 , then *Z*(*N* 1 ) <= *Z*(*N* 2 ). It is because we can never "lose" any trailing zero by multiplying by any positive number. We can only get new and new zeros. The function *Z* is very interesting, so we need a computer program that can determine its value efficiently.

**Program:**

#include <stdio.h>

#include <math.h>

int calculate(int n);

int main(void)

{

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* F CONTROL PROBLEM \*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\n");

int length;

printf("\n Enter no of times ");

scanf("%d", &length);

int answers[length];

int i, input;

printf("\n Enter the inputs \n");

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

{

scanf("%d", &input);

answers[i] =calculate(input);

}

printf("\n Outputs are: \n");

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

{

printf(" %d \n", answers[i]);

}

return 0;

}

int calculate(int n)

{

int values = 0;

if (n < 5)

return values;

int k = 1;

while ((int)floor(pow(5, k + 1)) < n)

++k;

int i;

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

{

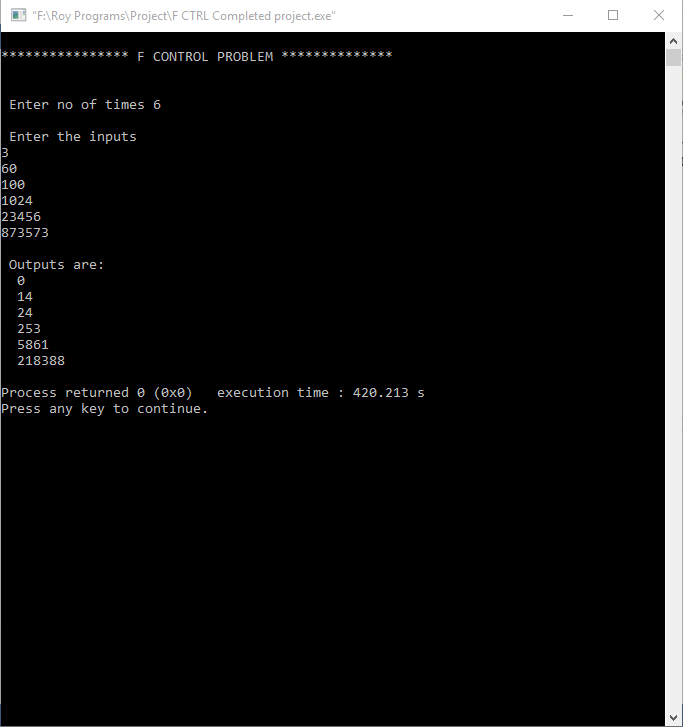
values += floor((n / pow(5, i)));

}

return values;

}

**Output:**

****

**Question 2:**

**SHPATH**

Given a list of cities. Each direct connection between two cities has its transportation cost (an integer bigger than 0). The goal is to find the paths of minimum cost between pairs of cities. Assume that the cost of each path (which is the sum of costs of all direct connections belongning to this path) is at 200000. The name of a city is a string containing characters a,...,z and is at most 10 characters long.

**Program :**

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

struct node

{

int adj;

int cost;

struct node \* next;

};

typedef struct node \* List;

struct graphentry

{

int n\_vertex;

List \*array;

};

typedef struct graphentry \*Graph;

struct tableentry

{

int known;

int dist;

int path;

};

struct city

{

char cit[10];

}ci[20];

typedef struct tableentry \* Table;

Graph gInit(int n);

Table tInit(int n,int s);

void readGraph(Graph g);

int findMinVertex(Table t,int n);

void dijkstra(Graph g,Table t);

void city(int);

int distance(Graph g,Table t,char [],char[],int n);

Graph gInit(int n)

{

int i;

Graph g=(Graph)malloc(sizeof(struct graphentry));

g->n\_vertex=n;

g->array=(List \*)malloc(sizeof(List)\*(n+1));

for(i=0;i<n+1;i++)

{

g->array[i]=(List)malloc(sizeof(struct node));

g->array[i]->next=NULL;

}

return g;

}

void readGraph(Graph g)

{

List t,temp;

int a,c,i,ch;

for(i=1;i<=g->n\_vertex;i++)

{

printf("\n Enter Data about city %s labeled as %d \n",ci[i].cit,i);

do

{

printf(" Enter 1 to add a route with continent %s ",ci[i].cit);

scanf("%d",&ch);

if(ch==1)

{

printf("Enter adjacent continent label & its traveling cost:");

scanf("%d%d",&a,&c);

t=g->array[i];

temp=(List)malloc(sizeof(struct node));

temp->adj=a;

temp->cost=c;

temp->next=t->next;

t->next=temp;

}

}while(ch==1);

}

}

Table tInit(int n,int s)

{

int i;

Table t;

t=(Table)malloc(sizeof(struct tableentry)\*(n+1));

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

{

t[i].known=0;

t[i].dist=9999;

t[i].path=0;

}

t[s].dist=0;

return t;

}

int findMinVertex(Table t,int n)

{

int minv=0;

int min=9999;

int i;

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

{

if(t[i].known==0)

{

if(min>t[i].dist)

{

min=t[i].dist;

minv=i;

}

}

}

return minv;

}

void dijkstra(Graph g,Table t)

{

int v,w,i;

for(i=1;i<=g->n\_vertex;i++)

{

v=findMinVertex(t,g->n\_vertex);

if(v<1)

break;

t[v].known=1;

List temp=g->array[v]->next;

while(temp!=NULL)

{

w=temp->adj;

if(t[w].known==0)

{

if(t[w].dist>temp->cost)

{

t[w].dist=temp->cost;

t[w].path=v;

}

}

temp=temp->next;

}

}

}

void display(Table t,int n)

{

int i;

char a[10];

printf("\n\n\t\t\t\t\t\t\t\t TABLE IS \n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\ncontinent no\t\tcontinent\t\tVisited\t\t\t\tDist\t\t\t Path\n");

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

{

printf("%d\t\t\t%s\t\t\t%d\t\t||\t\t%d\t\t||\t\t%d\n",i,ci[i].cit,t[i].known,t[i].dist,t[i].path);

printf("----------------------------------------------------------------------------------------------------------------\n");

}

}

void city(int n)

{

int i;

char a[10];

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

{

printf("\n Enter %d continent name ",i);

scanf("%s",ci[i].cit);

}

}

int distance(Graph g,Table t,char s[],char d[],int n)

{

int i,soc,j;

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

{

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

{

if((!strcmp(ci[i].cit,s))&&(!strcmp(ci[i].cit,s)))

{

soc=i;

}

}

t[i].dist+=t[i].dist;

}

return t[i].dist;

}

int main()

{

int n,st,sho;

char s[10],d[20],ch;

printf("\n\n\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\* SHIPPATH PROGRAM USING GRAPH DATA STRUCTURE \*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n\n");

printf("\n Enter no of Continents ");

scanf("%d",&n);

Graph g=gInit(n);

city(n);

readGraph(g);

printf("\n Enter Starting Contient ");

scanf("%d",&st);

Table t=tInit(n,st);

dijkstra(g,t);

display(t,n);

roy:

loose:

printf("\n Enter the ship start continent ");

scanf("%s",s);

printf("\n Enter the ship destination continent ");

scanf("%s",d);

if(!strcmp(s,d))

{

printf("\n You have entered Start and Destination same !!!...\n");

goto loose;

}

else

{

sho=distance(g,t,s,d,n);

}

if(sho==9999)

{

printf("\n Both Continents will not be visited because both are not connected !!!");

return 1;

}

else

{

printf("\n Minimum distance to reach %s from %s is \* %d units \* ",d,s,sho);

}

// printf("\n Do you want to find other path ? Y/N ");

//scanf("%c",&ch);

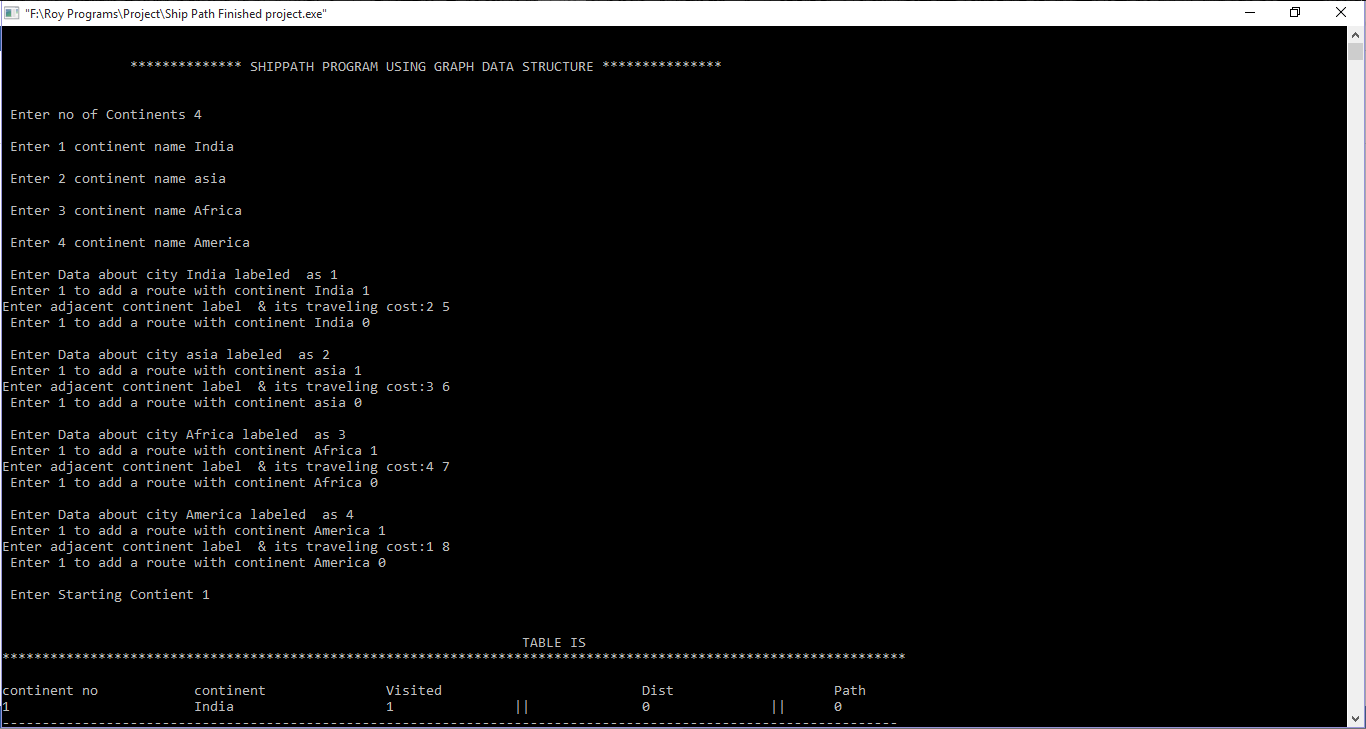
//if(ch=='y')

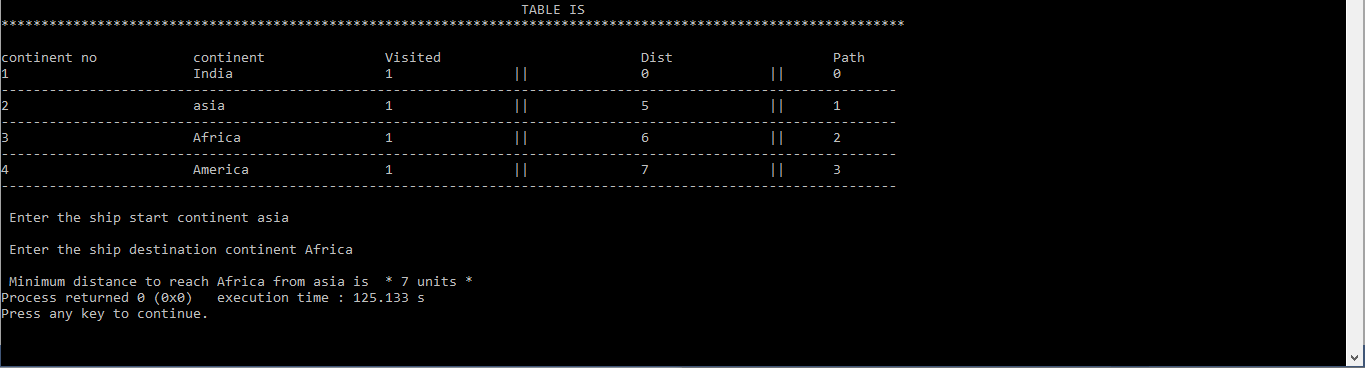
// goto roy;

return 0;

}

**Output:**

****

****

**Question 3:**

**MMIND**

If you want to buy a new cellular phone, there are many various types to choose from. To decide which one is the best for you, you have to consider several important things: its size and weight, battery capacity, WAP support, colour, price. One of the most important things is also the list of games the phone provides. Nokia is one of the most successful phone makers because of its famous Snake and Snake II. ACM wants to make and sell its own phone and they need to program several games for it. One of them is Master-Mind, the famous board logical game.

The game is played between two players. One of them chooses a *secret code* consisting of *P* ordered pins, each of them having one of the predefined set of *C* colours. The goal of the second player is to guess that secret sequence of colours. Some colours may not appear in the code, some colours may appear more than once.

The player makes guesses, which are formed in the same way as the secret code. After each guess, he/she is provided with an information on how successful the guess was. This feedback is called a *hint*. Each hint consists of *B* black points and *W* white points. The black point stands for every pin that was guessed right, i.e. the right colour was put on the right position. The white point means right colour but on the wrong position. For example, if the secret code is "white, yellow, red, blue, white" and the guess was "white, red, white, white, blue", the hint would consist of one black point (for the white on the first position) and three white points (for the other white, red and blue colours). The goal is to guess the sequence with the minimal number of hints.

The new ACM phone should have the possibility to play both roles. It can make the secret code and give hints, but it can also make its own guesses. Your goal is to write a program for the latter case, that means a program that makes Master-Mind guesses.

**Program:**

#include <stdio.h>

#include <string.h>

#include <stdarg.h>

#include <stdlib.h>

typedef struct set

{

int count;

void \*\*values;

} \*SR;

SR create(void \*values,...);//

int set\_count(SR this);

int contains(SR this, void \*value);

int rscore(SR set1, void \*value, int score);

int main(int argc,char \*\*argv )

{

int t = 0, n = 0 ,k = 0,q = 0, score = 0,i,c;

char ch;

int arr1[n];

int arr2[n];

printf("Please enter the number of test cases(between 1 to 100):");

scanf("%d",&t);

printf("\n");

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

{

//printf("Please enter values of P , C ,M values : ");

scanf("%i %i %i",&n, &k,&q);

printf("\n");

// printf("Enter the values of secret key ");

score = 0;

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

{

scanf("%d",&arr1[c]);

}

printf("\n");

// printf("Enter the values of guess ");

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

{

scanf("%d",&arr2[c]);

}

}

SR set1 = create(&arr1);

SR set2 = create(&arr2);

for ( i = 0; i < set2->count; i++)

{

void \*val = set2->values[i];

score = rscore(set1, val,score);

}

if ( score == set1->count)

printf("\n Congrats Yes You Won....");

else

printf("\n No You Are Cheating !!!!!");

printf("\n");

}

SR create(void \*values,...)//

{

SR set = calloc(1, sizeof(struct set));

if (values)

{

int count = 1;

va\_list args;

va\_start(args, values);

while (va\_arg(args, void \*))

{

count++;

}

va\_end(args);

set->count = count;

set->values = calloc(count, sizeof(void \*));

set->values[0] = values;

va\_start(args, values);

int i = 1;

void \*val;

while ((val = va\_arg(args, void \*)))

{

set->values[i++] = val;

}

va\_end(args);

}

return set;

}

int set\_count(SR this)

{

return this->count;

}

int contains(SR this, void \*value)

{

int i;

for ( i = 0; i < this->count; i++)

{

if (value == this->values[i])

return 1;

}

return 0;

}

int rscore(SR set1, void \*value, int score)

{

int t;

void \*val = value;

t=contains(set1, val);

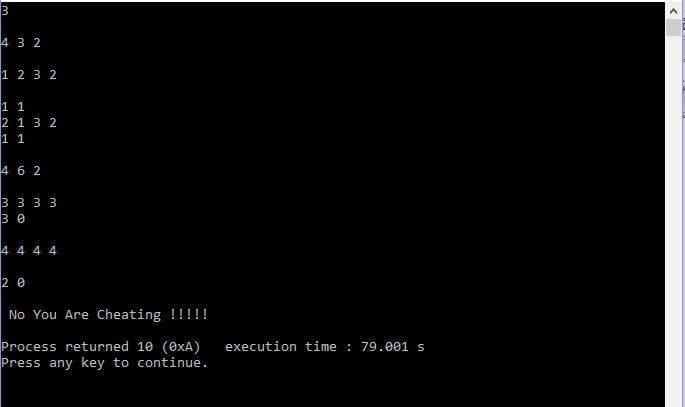
if (t==1)

score ++;

return score;

}

**Output:**

****