Siddaganga Institute of Technology, Tumkur – 572 103

Department of Computer Science and Engineering

Semester : III **Odd 2016-17** Subject : Data structures-3CCI01

Programming Assignment 1:

Given a list of student names and their marks in K subjects, find the average mark for every student. Also, print a list of toppers in each subject. (Assume there is only one topper).

Use the given structure Student (in C language). You should the implement following functions which are called in main():

- ***char * getTopper(struct Student s[], int nStudents, int subjectIndex, int nSubjects);***

```
/*
Return name of topper of a subject.
Arguments :
        array of Student records.
        number of Students (nStudents).
        index of subject for which function should find toper (subjectIndex).
        number of Subjects (nSubjects)
*/
```

- ***float getAvg(struct Student s, int nSubjects);***

```
/*
return avg mark of a student.
Arguments :
        a Student record s.
        number of Subjects (nSubjects)
*/
```

- ***void setValues(struct Student* s, const char * name, float marks[], int nSubjects);***

```
/*
set values in structure Student
Arguments :
        a pointer to a Student record (s).
        name of Student (name).
        mark of K subjects (marks).
        number of Subjects (nSubjects)
*/
```

Input:
Given a list of student names and their marks in K subjects, find the average mark for every student. Also, print a list of toppers in each subject. (Assume there is only one topper).

The first line contains the number of students (N) and the number of subjects (K). Following N lines contains name of student and the marks of K subjects.

Output
The first N lines of output should contain average marks of each student (Print 6 decimal places) . Following K lines should contain the name of the topper of each subject in the order in which the subjects are given.

Constraints
1 <= N <=100
1 <= K <=10
Assume that Name does not contain spaces and the maximum length of names of the students is 10 (excluding the NULL character at the end).
0 <= Mark <=100 and the marks are real numbers.

Solution:

```
/*

Date:  2-10-2016*/

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

struct Student{
   char name[11];
   float marks[10];
};
void setValues(struct Student*s,const char*name,float marks[],int nSubjects);
float getAvg(struct Student *s,int nSubjects);
char*getTopper(struct Student*s,int nStudents,int subjectIndex,int nSubjects);
int main()
{
   struct Student *s;
   unsigned n,k;
   int i;
enterNo:printf("Enter the no. of students\n");
   scanf("%u",&n);
   if(n<=1||n>=100){
      printf("No of Students limit ranges from 1 to 100\n");
      goto enterNo;
   }
enterSub: printf("Enter the no of subjects\n");
   scanf("%u",&k);
   if(n<=1||n>=10){
      printf("No of Subjects limit ranges from 1 to 10\n");
      goto enterSub;
   }

   //Allocating memory for the structure dynamically
   s=(struct Student*)calloc(n,sizeof(struct Student));
   for(i=0;i<n;i++){
   printf("Enter the name of the student %d : ",i+1);
   setValues(s,(s+i)->name,(s+i)->marks,k);
   }
   //Calculating Average of each student marks
   printf("\n\nThe average marks of the students are\n");
   for(i=0;i<n;i++){
   printf("%-10s :",(s+i)->name);printf("  %f\n\n",getAvg(s+i,k));
   }

   //Getting Topper]
   printf("\n ****The List of toppers in each subject is****\n\n");
   for(i=0;i<k;i++){
      printf("Sub %d : ",i+1);
      printf("%-10s\n",getTopper(s,n,i,k));

   }
free(s);
printf("Successful Termination.Press any button to exit\n");
getch();
```

```
   return 0;
}
void setValues(struct Student*s,const char*name,float marks[],int nSubjects){
   int i,j;
      scanf("%s",name);
      printf("Enter the marks of the student in %d subjects\n",nSubjects);
       for(j=0;j<nSubjects;j++)
      {
         printf("Sub %d:",j+1);
         scanf("%f",&marks[j]);
      }



}

float getAvg(struct Student *s,int nSubjects){
int i,sum=0;
for(i=0;i<nSubjects;i++)
   sum=sum+s->marks[i];
return sum/nSubjects;
}

char*getTopper(struct Student*s,int nStudents,int subjectIndex,int nSubjects)
{
   int i,largest=0;
   for(i=0;i<nStudents;i++)
   {
      if(s[i].marks[subjectIndex]>s[largest].marks[subjectIndex])
         largest =i;
   }

   return s[largest].name;
}
```

Programming Assignment 2:

You have to implement a matrix ADT using concepts of C taught in the lectures. The input matrices would be square matrices. The class must support the following functions:

1. *Matrix Addition*
2. *Matrix Subtraction*
3. *Matrix Multiplication*

The program should take as input: dimension of a square matrix N, two matrices of size N x N with integer values, and one operator symbol (+, - ,). It must perform the corresponding operation using member functions of Matrix class. You are supposed to implement the following functions of the class MatrixADT* ***matrixType add(matrixType M1, matrixType M2)****: The function should add matrices M1 and M2 and store the result in "resultMatrix". The function should return the "resultMatrix".*

matrixType subtract(matrixType M1, matrixType M2)*:*

The function should subtract matrix M2 from M1 and store the result in "resultMatrix". The function should return the "resultMatrix".

matrixType multiply(matrixType M1, matrixType M2)*:*

*The function should multiply matrices M1 and M2 and store the result in "resultMatrix". Be careful, it is M1*M2 and not M2*M1. The function should return the "resultMatrix".*

INPUT:

*In the first line, one integer which is the dimension of the matrix and one operator (one of +, - or *)*

Two NxN matrices one after the other, supplied one row at a time.

OUTPUT:

Resultant matrix after performing the specified operation, one row at a time. For subtraction, if A is the first matrix and B is the second matrix, perform A-B.

CONSTRAINTS:

The inputs will satisfy the following constraints:

1<=N<=10

There is no need to validate the value of N.

Solution:

```
/*Problem 2 ,
      Date:  2-10-2016*/
#include<stdio.h>
#include<stdlib.h>

typedef struct {
int arr[10][10];
int dimension;
}matrixType;

matrixType add(matrixType,matrixType);
matrixType subtract(matrixType,matrixType);
matrixType multiply(matrixType,matrixType);

int main(){
int k,i,j,n;char op;
matrixType *m=(matrixType*)malloc(4*sizeof(matrixType));
```

```
printf("Enter the dimensions of the square matrix\n");
scanf("%d",&n);

m->dimension=n;
for(k=0;k<2;k++){
printf("Enter the matrix %d elements\n",k+1);
for(i=0;i<m->dimension;i++){
   printf("Enter the elements of row %d\n",i+1);
   for(j=0;j<m->dimension;j++)
      scanf("%d",&((m+k)->arr[i][j]));
}
}

enterOpr:printf("Enter the operator as \n+ for addition\n- for subtraction\n* for
multiplication\n");
       scanf("%*c%c",&op);

switch(op){

      case '+': m[2]=add(m[0],m[1]);
         break;
      case '-': m[2]=subtract(m[0],m[1]);
         break;
      case '*':m[2]= multiply(m[0],m[1]);
         break;
      default:printf("Wrong operator entered\n");
            goto enterOpr;
}
printf("Displaying resultant matrix \n");
for(i=0;i<m->dimension;i++){
   for(j=0;j<m->dimension;j++)
      printf("%d\t",m[2].arr[i][j]);
      printf("\n");
}
return 0;
}

matrixType add(matrixType M1,matrixType M2){
int i,j,k;

matrixType result;
```

```
for(i=0;i<M1.dimension;i++)
   for(j=0;j<M1.dimension;j++)
      result.arr[i][j]=M1.arr[i][j]+M2.arr[i][j];
   return result;



}
matrixType subtract(matrixType M1,matrixType M2){
int i,j,k;

matrixType result;

for(i=0;i<M1.dimension;i++)
   for(j=0;j<M1.dimension;j++)
      result.arr[i][j]=M1.arr[i][j]-M2.arr[i][j];
   return result;



}
matrixType multiply(matrixType M1,matrixType M2){
int i,j,k,sum;

matrixType result;

for(i=0;i<M1.dimension;i++){

   for(j=0;j<M1.dimension;j++){
      sum=0;
      for(k=0;k<M1.dimension;k++)
         sum=sum+M1.arr[i][k]*(M2.arr[k][j]);
         result.arr[i][j]=sum;

   }



}
   return result;



}
```

Programming Assignment 3:

There are no constraints on the values of the entries of the matrices.

Write a program which does the following:
There are 3 tables in a room. Every table can hold any number of blocks stacked on each other with only one block touching the table. No two blocks are of the same size. At any point of time, the blocks must be placed in ALL tables in such a way that no smaller sized block will have a larger sized block on top of it. Initially all the blocks are placed on table number 1 in the correct fashion (biggest block at bottom and smallest block at the top). Our aim is to ***move*** *all the blocks from table number 1 to table number 3 such that at every step all the constraints are satisfied and the final order in table 3 is: smallest block must be at the top and the largest at the bottom and all the other blocks in between in increasing size. Tables 2 and 1 should not have any blocks on them at the end of the process.* ***You are allowed to move only one block at a time from any table to any other table.***
Write a program which takes number of blocks as input and outputs the number of moves required to do so.
Input: *a single integer indicating number of blocks*
Output:*a single integer indicating the number of moves required*
Constraints: *You can assume that the number of blocks will be atmost 20.*
Sample Test Cases

	Input	*Output*
Test Case 1	*9*	*511*
Test Case 2	*2*	*3*
Test Case 3	*3*	*7*
Test Case 4	*18*	*262143*
Test Case 5	*1*	*1*
Test Case 6	*3*	*7*
Test Case 7	*5*	*31*
Test Case 8	*7*	*127*
Test Case 9	*15*	*32767*
Test Case 10	*13*	*8191*
Test Case 11	*19*	*524287*

Solution:

```
/*Problem 3 ,
   Date:  2-10-2016*/


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

int TOH(int n,char from,char aux,char to);
int count=0; //counter variable to count the number of steps in the
int main()
{

   int n;

   for(;;){
       count=0;
  // printf("Enter the no of blocks \n");
  enterAgn: scanf("%d",&n);
```

```
   if(n>20){
      printf("Max no. of blocks exceeded\n");
      goto enterAgn;
   }


   printf(" %d\n",TOH(n,'1','2','3'));//'1'-table1 '2'-table2 '3'-table3
   }
   return 0;
}
int TOH(int n,char from,char aux,char to){

if(n==1){
   count++;
return count;
}

TOH(n-1,from,to,aux);

TOH(1,from,aux,to);

TOH(n-1,aux,from,to);
return count;

}
```

Programming Assignment 4:

Your college is organising a sporting event. The events are all done and now the task is to prepare the list of medal winners. The sports secretary for some reason assumed that the events will only have Gold and Silver medals but no Bronze medals. The contestants, especially the 3rd place winners were furious (understandably so!). To get things in control the secretary gives you the list of participants in an event and their standing in the event. Your task is to output the list of Bronze medalists in each of the events.

Each event has 10 participants.
***Input**: Each line has 10 numbers that indicate the time taken by each contestant to complete the task. Each task takes less than or equal to 30 seconds. If there are two bronze winners, print the one with the lower id.*
***Output**: The id of the contestant who secured the Bronze medal.*

***Example**:*
Input

16 12 34 11 10 5 3 1 15 21
Output
6
Explanation: The 1 in first line indicates that there is only one testcase. The third position is for the player who took 5 sec to finish the task. The player is in the 6th person in the input list. ***Note that the ids start at 1 and not at 0.***

Solution:

```
/*Problem 4,
      Date:  2-10-2016*/

#include<stdio.h>
#include<stdlib.h>
int getBronzeMedalist(int*a,int n);
int main(){
int *a = (int*)calloc(10,sizeof(int));
int e=1;
for(;;){
printf("\t\t*******\t\tEvent %d\t\t********\n\n",e);
printf("Enter the time(in sec) taken by the participants in order of their ID's \n");
int i;
for(i=0;i<10;i++)
   scanf("%d",&a[i]);
for(i=0;i<10;i++)
if(getBronzeMedalist(a,10)==a[i]) //return value(time duration of bronze medalist) is
compared with original array elements
   break;
printf("%d\n",i+1);

e++;
}
free(a);
return 0;
}
int getBronzeMedalist(int*a,int n){

   int i;
   int count =0;
   int b[n];
   for(i=0;i<n;i++) //copying array
      b[i]=a[i];
   sort(b,n);

   for(i=0;i<n;i++)     //ruling out repeatition factor of elements
      if(b[i]!=b[i+1]){
         count++;
         if(count==3)
         return b[i];
      }
}

void sort(int*b,int n){
int i,j,small=0,temp;
 for(i=0;i<n;i++){
      small=i;
      for(j=i+1;j<n;j++)
         if(b[small]>b[j])
            small=j;
      temp=b[small];
      b[small]=b[i];
      b[i]=temp; }
}
```