Lab 02 (Dated 25-09-2020)

1. Bag

Create a structure Bag. Bag has collection of different elements with possibility of repetition. The order of elements is the order in which elements are added in the bag except insert operation to insert some element in between. The member variables are integer pointer, current size and max size. You have to write add option (without check of duplication). You have to write insertion function to insert element at particular index. You have write remove function to remove element by shifting next elements on left side to fill the place of deleted element.

Finally write main to demonstrate your structure. Read a positive number that is max size of Bag. Next read another positive number that is no of elements to be added in the bag. Add elements and print elements on the screen in single line. Next read a number and remove if number exist in the bag. Again print remaining elements. Next read two integers, first is number and second is index. Insert number and again print all numbers.

Input Format

5

5

23 45 28 63 64

45

33 2

Output Format

23 45 28 63 64

23 28 63 64

23 28 33 63 64

Submissions:

39

Max Score:

10

Difficulty:

Medium

Rate This Challenge:

More

Admin Options

Edit Challenge

View Submissions

Solution:

```
struct Bag{
    int *data, currentSize, size;
    void set(const int SIZE){
        size = SIZE;
        currentSize = 0;
        data = new int[size];
    }
    bool add(const int ELEMENT){
        if (currentSize == size)    return false;
        data [ currentSize++ ] = ELEMENT;
        return true;
    }
```

```
    bool remove(const int ELEMENT){
        int i;
        for (i=0 ; i < currentSize && data[i] != ELEMENT; i++);
        if (data[i==ELEMENT]){
            currentSize--;
            for ( ; i < currentSize;i++)
                data[i] = data[i+1];
            return true;
        }
        return true;
    }
    bool insert(const int ELEMENT, const int INDEX){
        int i;
        if (currentSize>=size)  return false;
        for (i=currentSize ; i >= INDEX ; i--)
            data[i+1]=data[i];
        data[INDEX]=ELEMENT;
        currentSize++;
        return true;
    }
    void show() const{//Function can't change variables of structure
        if (currentSize == 0 )
            cout << "Empty Bag";
        for (int i=0;i<currentSize;i++)
            cout << data[i] << ' ';
        cout << '\n';
    }
    void free(){
        delete []data;
    }
};

int main(){
    struct Bag bag;
    int value, i , n;
    cin >> n;
    bag.set(n);
    cin >> n;
    for (i=0;i<n;i++){
        cin >> value;
        bag.add(value);
    }
    bag.show();
    cin >> value;
    bag.remove(value);
    bag.show();
    cin >> value >> n;
    bag.insert(value, n);
    bag.show();
    bag.free();
    return 0;
```

```
}
```

2. Count Common Row Wise

Create a structure Matrix having member interger type variables, 2d pointer, rows, columns. Write function to initialize and remove dyanmic memory. Write functions to read and print values. Print function is to print all elements in matrix form. Read function will read values and place into Matrix. Write function to multiply ith row of matrix with some element, note here i starts from 1 not from 0. WWrite function to interchange ith column with jth column.

Finally, writ main to demonstrate Matrix. Read rows & columns and initialize Matrix. Next read (rows x columns) elements and place in matrix row wise. Print matrix. Next read two integers. First is row number (again starting row is 1, be careful), second is number for multiplication. Multiply all elements of specific row with the number. Lastly, read two more integers representing column numbers. Interchange columns accordingly. Final print matrix again. See Input Output carefully for further understanding.

Input Format

```
3 2
2 6 4
4 3 1
2 2
1 3
```

Output Format

```
2 6 4
4 3 1
4 6 2
2 3 8
```

Solution:

```
#include <iostream>

using namespace std;

struct Matrix{
    int **data, rows, columns;
    void set(const int ROWS, const int COLUMNS){
        rows = ROWS;
        columns = COLUMNS;
        data = new int*[ROWS];//memory of 1d pointers
        for (int i=0;i<ROWS;i++)
            data[i] = new int [COLUMNS];
    }
    void show(){
        int i,j;
        for (i=0;i<rows;i++){
            for (j=0;j<columns;j++)
                cout << data[i][j] << ' ';
            cout << '\n';
        }
    }
    void readValues(){
        int i,j;
        for (i=0;i<rows;i++)
            for (j=0;j<columns;j++)
                cin >> data[i][j];
    }
    void multiplyIthRowWithK(const int I, const int K){
        int j;
        for (j=0;j<columns;j++)
```

```
            data[I-1][j] *= K ;
    }
    void interchangeIthColumnWithKthColumn(const int I, const int K){
        int i, temp;
        for (i=0;i<rows;i++){
            temp = data[i][I-1] ;
            data[i][I-1] = data[i][K-1];
            data[i][K-1] = temp;
        }
    }
    void free(){
        for (int i=0;i<rows;i++)
            delete []data[i];
        delete data;
    }
};

int main(){
    srand(time(0));
    Matrix m;
    int r, c;
    cin >> r >> c;
    m.set(r,c);
    m.readValues();
    m.show();
    cin >> r >> c;
    m.multiplyIthRowWithK(r,c);
    cin >> r >> c;
    m.interchangeIthColumnWithKthColumn(r,c);
    m.show();
    m.free();
    return 0;
}
```

3. Find Sequence

A 2D array has elements out of order. You have to find the correct order of elements. Note that you don't have to give sorted list rather you have to give indexes of elements so that one can read them in order. For example, consider following 2D array of size 2 x 4:

28 45 48 36 49

50 51 38 39 23

The correct order is 2,5,1 - 1,1,1 - 1,4,1 - 2,3,2 - 1,2,2 - 1,5,3. Here first element is in second row, fifth column, second element is in first row, first column, third element is in first row, fourth column, then there are two elements starting from second row, third column, again there are two elements starting from first row, second column and lastly there are three elements. First element is in first row fifth column and two elements are in start of second row. You have to read rows, columns. Next read values row x columns. You have to print correct sequence of elements in triplet form. See output format carefully.

Input Format

2 5
28 45 48 36 49
50 51 38 39 23

Output Format

```
2 5 1
1 1 1
1 4 1
2 3 2
1 2 2
1 5 3
```

Solution:

```
struct Matrix{
    int **d, rows, columns;
    void init(int r, int c){
        rows=r, columns=c;
        int i;
        d = new int* [rows];
        for (i=0;i<rows;i++)
            d[i] = new int [columns];
    }
    void read(){
        int i,j;
        for (i=0;i<rows;i++)
            for (j=0;j<columns;j++)
                cin>>d[i][j];
    }
    void print(){
        int i,j;
        for (i=0;i<rows;i++){
            for (j=0;j<columns;j++)
                cout << d[i][j] << ' ';
            cout << '\n' ;
        }
    }
    void copyElementIntoArray(int *a){
        int i,j, k=0;
        for (i=0;i<rows;i++)
            for (j=0;j<columns;j++)
                a[k++]=d[i][j];
    }
    void sort(int *a, const int SIZE){
        int i, j, value;
        for (i=1;i<SIZE;i++){
            value = a[i];
            for (j=i-1;j>=0 && a[j]>value;j--)
                a[j+1]=a[j];
            a[j+1]=value;
        }
    }
    void findPosition(const int ELEMENT, int &i, int &j){
        for (i=0;i<rows;i++)
            for (j=0;j<columns;j++)
                if (d[i][j]==ELEMENT)
```

```
                    return;
    }
    void findAndPrintSequence(){
        int size = rows * columns;
        int *a = new int[size];
        copyElementIntoArray(a);
        sort(a, size);
        int r=0, c=0, i, j, count;
        for (int k=0 ; k<size ; k++){
            findPosition(a[k], r, c);
            i = r, j = c;
            count=1;
            while (true){
                if (i==rows-1 && j==columns-1)  break;
                if (j==columns-1 )
                    i++, j=0;
                else
                    j++;
                if (d[i][j]==a[++k]) count++;
                else { k--; break;}
            }
            cout << r+1 << ' ' << c+1 << ' ' << count << '\n';
        }
    }

};
int main() {
    Matrix m;
    int r, c;
    cin >> r >> c;
    m.init(r, c);
    m.read();
    m.findAndPrintSequence();
    return 0;
}
```