Ch10

1. Reread the code in Display 10.9. Then, write a class **TwoD** that implements the two-dimensional dynamic array of **doubles** using ideas from this display in its constructors. You should have a private member of type pointer to **double** to point to the dynamic array, and two **int** (or **unsigned int**) values that are **MaxRows** and **MaxCols**.

   You should supply a default constructor for which you are to choose a default maximum row and column sizes and a parameterized constructor that allows the programmer to set maximum row and column sizes.

   Further, on should provide a **void** member function that allows setting a particular row and column entry and a member function that returns a particular row and column entry as a value of type **double**.

   Remark: It is difficult or impossible (depending on the details) to overload [ ] so it works as you would like for two-dimensional arrays. So simply use accessor and mutator functions using ordinary function notation.

   Overload the + operator as a friend function to add two two-dimensional arrays. This function should return the **TwoD** object Whose i^{th} row, j^{th} column element is the sum of the i^{th} row, j^{th} column element of the left-hand operand **TwoD** object and the i^{th} row, j^{th} column element of the right-hand operand **TwoD** object. Provide a copy constructor an overloaded operator =, and a destructor. Declare class member functions that do not change the data as **const** members.

```cpp
#include <iostream>
using namespace std;

typedef int* IntArrayPtr;

int main(){
    int d1, d2;
    cout<<"Enter the row and column dimensions of the array:\n";
    cin>>d1>>d2;

    IntArrayPtr *m = new IntArrayPtr[d1];
    int i,j;
    for (i=0; i<d1; i++)
        m[i] = new int[d2];
    //m is now a d1-by-d2 array.

    cout<<"Enter "<<d1<<" rows of "<<d2<<" integers each:\n";
    for(i=0; i<d1; i++)
        for(j=0; j<d2; j++)
            cin>>m[i][j];

    cout<<"Echoing the two dimensional array:\n";
    for(i=0; i<d1; i++){
        for(j=0; j<d2; j++)
            cout<<m[i][j]<<" ";
        cout<<endl;
    }

    for(i=0; i<d1; i++)
        delete[] m[i];
    delete[] m;

    return 0;
}

```

Code in Display 10.9

```
Enter the row and column dimensions of the array:
3 4
Enter 3 rows of 4 integers each:
1 2 3 4
5 6 7 8
9 0 1 2
Echoing the two dimensional array:
1 2 3 4
5 6 7 8
9 0 1 2
```

Result of the code in Display 10.9

3. Write a program that accepts a line of text input from the user and check if the number of words counted from both end of the text is equal. Your program should. work by using two pointers. The 'head' pointer should be set to the address of the first character in the string, and the 'tail' pointer should be set to the address of the last character in the string (i.e., the character before the terminating **null**). The program should keep incrementing the "head" pointer and keep decrementing the "tail" pointer until the pointers reach the other end. On finding whitespace characters while going across the string, increment the word counts and compare them at the end.

4. Create a class named Subscriber that has three member variables:

name	-	A string that stores the name of the subscriber.
numChannels	-	An integer that tracks how many channels the subscriber sub-scribes to.
channelList	-	A dynamic array of strings used to store the names of the channels the subscriber subscribes to.

Write appropriate constructor(s), mutator, and accessor Functions for the class along with the following:

- A function that inputs all values from the user, including the list of channel names. This function will have to support input for an arbitrary number of channels.
- A function that outputs the name and list of all channels.
- A function that resets the number of channels to 0 and the **channelList** to an empty list.
- An overloaded assignment operator that correctly makes a new copy of the list of channels.
- A destructor that releases all memory that has been allocated.

Write a main function that test all your functions.

8. Create a class called Television that has member variables

displayType	-	A string that stores the television's display type.
dimension	-	A variable of type double that stores the dimension of the television in inches.
connectivitySupport	-	A dynamic array of strings that stores the different connectivity modes supported by the television.

The class should have mutator and accessor functions to set and get the above member variables. Include a constructor that takes arguments of type **string, double**, and an array of strings to assign the three member variables. Also include a copy constructor.

Embed your class in a test program that reads the input from the user to set **displayType, dimension,**

and **connectivitySupport** values by default for a television. Your program should then read in input for the number of televisions. Your program should create as many **Television** objects as required using the copy constructor, and ask the user if customization is required for any television. If so, use the mutator methods to set the values accordingly.