Question4.cpp

```cpp
//<----Lab 01 - Arrays and Dynamic Memory Allocation---->

//Q4. Using the abstract data Type of a Matrix given below, write a program that
// 1. Input a 4*3 matrix from user in 2D array
// 2. Map this array in 1D array using Row major order
// 3. Input second matrix of 3*4 in 2D array
// 4. Map this array in 1D array using Row major order.
// 5. Now perform matrix multiplication on these 1D arrays
// 6. Save the result back in a 2D array.

// Implement this question for any number of rows and columns using class “matrix”.

#include<iostream>
using namespace std;
class matrix{
    int **p;
    int r;
    int c;
    int *rowmajor;
    int *multiply1D;
    public:

        matrix(int row, int col);
        void disp2D(){
         cout<<"2D matrix element\t";
         for(int i=0;i<r;i++){
          for(int j=0;j<c;j++){
           cout<<p[i][j]<<" ";
          }
          cout<<endl;
         }
        }
        void dispRowMajor(){
         cout<<"row major order matrix"<<endl;
         for(int i=0;i<r*c;i++){
          cout<<rowmajor[i]<<" ";
         }
         cout<<endl;
        }
        void Multiply_rowMajor(matrix & x){
         if(c!=x.r){
          cout<<"multiplication not possible";
          return;
         }
         for(int i=0;i<r;i++){
          for(int j=0;j<c;j++){
           multiply1D[i*x.c+j]=0;
           for(int k=0;k<c;k++){
            multiply1D[i*x.c+j]+=rowmajor[i*c+k]*x.rowmajor[k*x.c+j];
           }
          }
         }
        }
```

```
54          void rowMajor_2D(){
55           int C = sizeof(multiply1D)/r;
56           cout<<"result of 2D in row major"<<endl;
57           for(int i=0;i<r;i++){
58            for(int j=0;j<C;j++){
59             cout<<multiply1D[i*c+j]<<" ";
60            }
61            cout<<endl;
62           }
63          }
64          ~matrix(){
65           for(int i=0;i<r;i++){
66            delete[] p[i];
67           }
68           delete[] p;
69           delete[] rowmajor;
70           delete[] multiply1D;
71          }
72  };
73
74  int main()
75  {
76      matrix a(4,3);
77      matrix b(3,4);
78      a.disp2D();
79      a.dispRowMajor();
80      b.disp2D();
81      b.dispRowMajor();
82      a.Multiply_rowMajor(b);
83      a.rowMajor_2D();
84  }
85
86  matrix::matrix(int row,int col){
87
88   r=row;
89   c=col;
90   p = new int*[r];
91    rowmajor = new int[r * c];
92      multiply1D = new int[r * c];
93
94      for (int i = 0; i < r; i++) {
95          p[i] = new int[c];
96          for (int j = 0; j < c; j++) {
97              p[i][j] = (i + j);
98              rowmajor[i * c + j] = p[i][j];
99          }
100     }
101 }
102
103
104 //First different approach
105 //#include<iostream>
106 //using namespace std;
107 //
108 //class matrix{
109 //  int **p;
```

```
110 //  int r;
111 //  int c;
112 //  int *rowmajor;
113 //  int *multiply1D;
114 //  public:
115 //  matrix(int row, int col);
116 //  // Constructor
117 //  void disp2D(){
118 //      for(int i=0;i<r;i++){
119 //          for(int j=0;j<c;j++){
120 //              cout<<p[i][j]<<"\t";
121 //          }
122 //          cout<<endl;
123 //      }
124 //  }
125 //  // displays the elements of **p
126 //  void dispRowMajor(){
127 //      int arr[r*c],x=0;
128 //      for(int i=0;i<r;i++){
129 //          for(int j=0;j<c;j++){
130 //              arr[x]=p[i][j];
131 //              x++;
132 //          }
133 //      }
134 //      for(int i=0;i<r*c;i++){
135 //          cout<<arr[i]<<endl;
136 //      }
137 //  }
138 //  // converts 2D into 1D using row major
139 //  //and displays the elements Row Major Order Matrix
140 //  void Multiply_rowMajor(matrix & x){
141 //      int k=0;
142 //      int A[r * c],B[x.r * x.c];
143 //      for(int i=0;i<r;i++){
144 //          for(int j=0;j<c;j++){
145 //              A[k]=p[i][j];
146 //              k++;
147 //          }
148 //      }
149 //      k=0;
150 //      for(int i=0;i<x.r;i++){
151 //          for(int j=0;j<x.c;j++){
152 //              B[k]=x.p[i][j];
153 //              k++;
154 //          }
155 //      }
156 //      int res[r * x.c];
157 //
158 //      int i, j;
159 //
160 //      for (i = 0; i < r; i++) {
161 //          for (j = 0; j < x.c; j++) {
162 //              res[j + i * x.c] = 0;
163 //
164 //              for (k = 0; k < c; k++) {
165 //                  res[j + i * x.c] += A[k + i * c] * B[j + k * x.c];
```

```
//              }
//          }
//      }
//      cout<<endl<<endl;
//      for (i = 0; i < r * x.c; i++)
//      cout<<res[i]<<endl;
//  }
//  void rowMajor_2D(){
//      return;
//  }
//  // Maps the elements stored in row major order to
//  // the 2D array and print the results
//  ~matrix(){
//      delete p;
//  }
//  // Destructor
//};
//
//int main()
//{
//  matrix a(4,3);
//  matrix b(3,4);
//  a.disp2D();
//  a.dispRowMajor();
//  b.disp2D();
//  b.dispRowMajor();
//  a.Multiply_rowMajor(b);
//  a.rowMajor_2D();
//}
//
//matrix::matrix(int row,int col)
//{
//  r=row;
//  c=col;
//  p = new int*[r];
//  for(int i=0;i<r;i++)
//  {
//      p[i]=new int[c];
//      for(int j=0;j<c;j++)
//      p[i][j]=(i+j);
//  }
//  // CODE FOR STORING DATA FROM
//  // **P TO *rowmajor ROW MAJOR
//}
```