3.1

a) The address of m[1][3] will be &m[1][3]

b) The address of m[0] is also the address of m[0][0], which is &m[0][0].

c) The address of array m is also the address of the first element of the array, which is m[0][0]. Thus it will also be &m[0][0].

d)

The code will be:

int(\*p)[6] = a;

cout << \*(\*(p + 1) + 3) << endl;

3.2

a) The 1st line double \*a[10] will be allocated with 4x10 = 40 bytes, but then they are de-allocated when the double \*a[10] was initialized

The 2nd line will be allocated 4x5 = 20 bytes.

The 3rd line will be allocated 0 bytes.

b) How the memory was allocated: From a[0] to a[4], we’ll ask for 4 more bytes in each of these elements.

c) void release(double\*\* a) {

for (int i = 0; i < 5; i++) {

delete[](a + i);

}

3.3

a) typedef void (\*process)( );

b) typedef int (\*power)(int x, int n);

c) typedef int (\*inputArr)(int& n);

d) typedef void (\*printArr)(int a[], int n);

e) typedef Fraction (\*add)(Fraction f1, Fraction f2)

3.4

Header file:

#ifndef \_HEADER\_H\_

#define \_HEADER\_H\_

#include <iostream>

using namespace std;

bool isPrime(int &n);

void Input(int& rows, int& cols, int\*\*& a);

void Output(int& rows, int& cols, int\*\*& a);

#endif

Header.cpp:

#include "Header.h"

bool IsPrime(int &n)

{

if (n <= 1)

return false;

for (int i = 2; i <= sqrt(n); i++)

if (n % i == 0)

return false;

return true;

}

void Input(int& rows, int& cols, int\*\*& a) {

for (int i = 0; i < rows; i++) {

a[i] = new int [cols];

for (int j = 0; j < cols; j++) {

cout << "Enter a[" << i << "][" << j << "]: ";

cin >> a[i][j];

}

cout << '\n';

}

}

void Output(int& rows, int& cols, int\*\*& a) {

for (int i = 0; i < rows; i++) {

for (int j = 0; j < cols; j++) {

if (IsPrime(a[i][j]) == true)

cout << a[i][j] << " ";

}

cout << '\n';

}

}

Main.cpp:

#include "Header.h"

using namespace std;

int main() {

int rows;

int cols;

int x = 0;

cout << "Enter the numbers of rows and columns: ";

cin >> rows >> cols;

int\*\* a = new int\* [rows];

Input(rows, cols, a);

Output(rows, cols, a);

for (int i = 0; i < rows; i++) {

delete[] a[i];

}

delete[] a;

return 0;

}