Name: ______________________________

Write all of your responses on these exam pages. If you need extra space please use the backs of the pages.

1 Short Answer

1. (*5 Points*) Write a function named `poparr` that takes in a single integer parameter named `size`. The function will create a dynamic array of integers, populate the array with random numbers between 1 and 100 inclusively, and finally return a pointer to the array.

2. (*7 Points*) Write a segment of code that will. Create a dynamic array of doubles of size 1000, call it A. Populate the array with random numbers between 0 and 1. Print out the array to the screen in one line of output. Then shift all of the elements of the array one entry to the left putting a 0 in the right most entry. Finally clear the array from memory. **All of this is to be done using pointer arithmetic for array access, (no A[i]).**

3. (*10 Points*) Given the following class. In the specification, add in the prototypes for two functions, add and dot. The add will return a vec3 type and the dot will return a double. Then implement these two functions outside the specification.

   The add will add the x values of the two objects, the y values of the two objects, and the z values of the two objects. The dot will multiply the x values, multiply the y values, and multiply the z values, and then add them all up. So if $a = (1, 2, 3)$ and $b = (4, 5, 6)$ then the addition will return the vec3 of $(5, 7, 9)$ and the dot will calculate $1 \cdot 4 + 2 \cdot 5 + 3 \cdot 6 = 32$, and return the 32. So the code on the left will produce the output on the right.

```
vec3 a(1, 2, 3);
vec3 b(4, 5, 6);
vec3 c = a.add(b);
a.print();
b.print();
c.print();
double d = a.dot(b);
cout << d << endl;
```

```
1 2 3
4 5 6
5 7 9
32
```

```
class vec3 {
private:
    double x, y, z;

public:
    vec3(double xv = 0, double yv = 0, double zv = 0) {
        x = xv; y = yv; z = zv; }

    void print() { cout << x << " " << y << " " << z << endl; }

    // Prototypes/specifications for add and dot




};

// Implementations of add and dot
```

4. (*12 Points*) Answer the following questions.

   (a) Assuming that ptr is a pointer to an int, what happens when you add 4 to ptr?

   (b) What is a default constructor?

   (c) Describe the difference between an instance member variable and a static member variable.

   (d) In what circumstances is the copy constructor called?

   (e) What is passed to the parameter of a class's operator= function? Specifically, if `A` and `B` are objects of this class, in the expression `A = B;` which gets sent to the parameter?

   (f) Why shouldn't a class's overloaded = operator be implemented with a void operator function?

2 Program Trace

1. (*7 Points*) Write the output of the following program given the inputs specified.

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

int main() {
    int A[] = { 4, 7, 13, 21, 42, 8, 25, 7, 9 };
    int *p, *q;

    p = A;
    p += 3;
    cout << *p << endl;
    q = p--;
    cout << *q << " " << *p << endl;
    cout << *p++ << endl;
    cout << *p << endl;
}
```

   (a) Output:

   (b) Is there a memory leak in this program or other memory related error? If so, where and why?

2. (*7 Points*) Write the output of the following program given the inputs specified.

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

int main() {
    int *A = new int[]{ 4, 7, 13, 21, 42, 8, 25, 7, 9 };
    int *p, *q;

    int a = 9;
    p = &A[3];
    q = A + a - 2;

    while (p++ < q)
        cout << *p << " ";
    cout << endl;

    for (int *p = A; p < &A[5]; p++)
        *p += *++p;

    for (int i = 0; i < a; i++)
        cout << A[i] << " ";
    cout << endl;

    int *c = A + 3;
    cout << *c << endl;
    delete c;
    int *d = A;
    delete d;

    return 0;
}
```

(a) Output:

(b) Is there a memory leak in this program or other memory related error? If so, where and why?

3. (*10 Points*) Write the output of the following program.

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

class Thing1 {
private:
    int a, b;

public:
    Thing1(int x = 1, int y = 2) {a = x; b = y;}
    void set(int x = 1, int y = 2) {a = x; b = y;}
    int geta() {return a;}
    int getb() {return b;   }
    void print() {cout << a << " " << b << endl;}
};

class Thing2 {
private:
    int a, b, c, *A;

public:
    Thing2(int x = 1, int y = 2, int z = 3) {
        a = x; b = y; c = z; A = new int[c];
        for (int i = 0; i < c; i++)
            A[i] = (i % a) + (i % b);
    }

    int geta() {return a;}
    int getb() {return b;}
    int getc() {return c;}
    int geta(int i) {return A[i];}
    void print() {
        cout << a << " " << b << " " << c << endl;
        for (int i = 0; i < c; i++)
            cout << A[i] << " ";
        cout << endl;
    }
};

int main() {
    Thing1 t1;
    Thing2 t2(4, 5, 6);
    Thing1 t3(7, 11);

    t1.print();
    t2.print();
    t3.print();

    cout << t1.geta() << endl;
    cout << t2.geta() << endl;
    cout << t3.getb() << endl;
    cout << t2.getb() << endl;
    cout << t2.geta(4) << endl;
    cout << t2.geta(5) << endl;
    cout << t2.geta(t1.getb()) << endl;

    return 0;
}
```

Output:

Is there a memory leak in this program or other memory related error? If so, where and why?

3 Coding

This exercise is to write a complete class structure named `Array2D` which has the following properties.

- The data is be in the private section, all functions are public. The data will be the array of pointers to integers, rows that will store the number of rows the array is storing and cols will be the number of columns the array is storing. The array of pointers will point to each row of the 2-D array. Each row will be a one-dimensional array of integers that is of the length of the number of columns of the array.
- The constructor will take in 3 parameters, the r, c, and defval. Have default values for each of these. r is the number of rows which should default to 3, c is the number of columns which should also default to 3, and defval is the value that the initial array will be populated with defaulted to 0. So the default constructor will create a 3×3 array populated with 0. In the constructor, make sure that the number of rows and the number of columns are not less than 1, so if r or c have values less then one, reset them to one before allocating memory.
- The destructor will free the memory allocation made for the array.
- display: Will simply print the array to the console. Don't worry about the columns lining up, just print each row on its own line with a space or two (or a tab if you would like) between each of the entries on the row.
- getRows: Will simply return the number of rows being stored in the array.
- getCols: Will simply return the number of columns being stored in the array.
- set: Will take in three parameters, the first is the row index, the second is the column index, and the third is the value to be stored in that position in the array. If the index is out of the allocated memory bounds the program should print the message `Index out of bounds.` to the console and not update the array in any way.
- get: Will take in two parameters, the first is the row index, and the second is the column index. The function will return the value in that cell of the array. If the index is out of the allocated memory bounds the program should print the message `Index out of bounds.` and just return 0.
- Transpose: Will take no parameters and not return anything. It will replace the array with its transpose. The transpose of an array is where the rows and columns are interchanged. So the first row becomes the first column of the transpose, second row to second column, and so on. For example, the transpose of the array

$$\begin{array}{cccc} 1 & 4 & 7 & 10 \\ 2 & 5 & 8 & 11 \\ 3 & 6 & 9 & 12 \end{array} \quad \text{is} \quad \begin{array}{ccc} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\ 10 & 11 & 12 \end{array}$$

- RowSums: Takes in no parameters and returns a pointer to a one-dimensional array that holds the sum of all the values on each row. For example, for the array

$$\begin{array}{cccc} 1 & 4 & 7 & 10 \\ 2 & 5 & 8 & 11 \\ 3 & 6 & 9 & 12 \end{array}$$

  the row sums would be an array with three entries: 22, 26, 30.

- ColumnSums: Takes in no parameters and returns a pointer to a one-dimensional array that holds the sum of all the values on each column. For example, for the array

$$\begin{array}{cccc} 1 & 4 & 7 & 10 \\ 2 & 5 & 8 & 11 \\ 3 & 6 & 9 & 12 \end{array}$$

  the column sums would be an array with four entries: 6, 15, 24, 33.

Some sample code and its output are below.

Sample Code Segment

```
Array2D A(3, 5, -1);

A.display();
cout << endl;

for (int i = 0; i < A.getRows(); i++)
    for (int j = 0; j < A.getCols(); j++)
        A.set(i, j, rand() % 10);

A.display();
cout << endl;

A.set(10, 3, -15);
A.display();
cout << endl;

A.set(1, 3, -15);
A.display();
cout << endl;

cout << A.get(2, 2) << endl;
cout << A.get(2, 20) << endl;
cout << endl;

A.Transpose();
A.display();
cout << endl;

int *rowsums = A.RowSums();
int *colsums = A.ColumnSums();

for (int i = 0; i < A.getRows(); i++)
    cout << rowsums[i] << " ";
cout << endl;

for (int j = 0; j < A.getCols(); j++)
    cout << colsums[j] << " ";
cout << endl;
```

Output

```
-1 -1 -1 -1 -1
-1 -1 -1 -1 -1
-1 -1 -1 -1 -1

4 3 1 3 5
3 4 5 8 1
9 3 6 5 2

Index out of bounds.
4 3 1 3 5
3 4 5 8 1
9 3 6 5 2

4 3 1 3 5
3 4 5 -15 1
9 3 6 5 2

6
Index out of bounds.
0

4 3 9
3 4 3
1 5 6
3 -15 5
5 1 2

16 10 12 -7 8
16 -2 25
```

1. (*9 Points*) Write the specification for the Array2D class. That is, what you would put in the Array2D.h file, you do not need to guard it here. No inline code at all.

2. (*5 Points*) Write the constructor.

3. (*5 Points*) Write the destructor.

4. (*3 Points*) Write the display function.

5. (*2 Points*) Write the getRows function.

6. (*2 Points*) Write the getCols function.

7. (*5 Points*) Write the set function.

8. (*5 Points*) Write the get function.

9. (*6 Points*) Write the Transpose function.

10. (*5 Points*) Write the RowSums function.

11. (*5 Points*) Write the ColumnSums function.