CS 7B - Spring 2018 - Final Exam

Write your responses to following questions on this paper, or attach extra, as needed. Use complete sentences where appropriate and write out code using proper style and syntax.

1. Suppose that the variable `x` is stored in the memory address `0x28ff04`. What will the following C++ code print (assume it doesn't crash)?

```
#include <iostream>

int main() {
    int x = 17;
    int* y = &x;
    int& xref = x;
    std::cout << ++x << std::endl;
    std::cout << *y << std::endl;
    std::cout << ++xref << std::endl;
    std::cout << &xref << std::endl;
    std::cout << ++y << std::endl;
    std::cout << *y << std::endl;
}
```

   Why might it crash?

2. Describe the output you might expect from the following program.

```
#include <iostream>

int main() {
    int x = 65;
    int* y = &x;
    for(int i = 0; i < 100; ++i) {
        std::cout << (char)*(y++);
    }
}
```

   Why is this program more likely to crash than the program in question # 1?

3. Compare and contrast the code below with that of problem # 2. Why is the cast needed here and why was it not needed in # 2?

```
#include <iostream>

int main() {
    int x = 97;
    void* y = &x;
    for(int i = 0; i < 100; ++i) {
        std::cout << *reinterpret_cast<char*>(y++);
    }
}
```

4. Consider the `Matrix` class partially defined below.

```
class Matrix {
public:
    Matrix(int rows, int cols);
    Matrix(const Matrix& rhs);
    Matrix& operator=(const Matrix& rhs);
    double& operator()(int i, int j);
    const double operator()(int i, int j) const;
    void print() const;
private:
    int mRows;
    int mCols;
    int mSize;
    double* mData;
};
Matrix::Matrix(int rows, int cols)
: mRows(rows), mCols(cols), mSize(rows * cols), mData(new double[mSize]) { }
Matrix::Matrix(const Matrix& rhs)
: mRows(rhs.mRows), mCols(rhs.mCols), mSize(rhs.mSize)
{   mData = new double[mSize];
    std::memcpy(mData,rhs.mData,rhs.mSize * sizeof(double));
}
Matrix& Matrix::operator=(const Matrix& rhs) {
    if(&rhs == this) {
        return *this;
    }
    if(mSize == rhs.mSize)
        std::memcpy(mData,rhs.mData,mSize * sizeof(double));
    else {
        delete[] mData;
        mData = new double[rhs.mSize]();
        std::memcpy(mData,rhs.mData,mSize * sizeof(double));
    }
    mRows = rhs.mRows;
    mCols = rhs.mCols;
    mSize = rhs.mSize;
    return *this;
}
```

(a) On which line is the prototype for the copy constructor declared?

(b) As it is, the constructor allocates memory for a Matrix, but does not initialize it. Where would you add code to intialize all matrix elements to 0 and what code would you add?

(c) How much of which part of memory would be allocated by the declaration `Matrix M(3,4)`?

(d) A prototype for the overloaded parentheses operator is declared on line 6. Write a definition for this function which returns the element in the `i`th row and `j`th column of the matrix.

(e) Write a definition for the class' `print()` method that uses the parentheses operator to access the element in row `i` and column `j`. Note that the "`const`" modifier of the method promises not to change any of the class' data members. Thus you will need a second definition for the overloaded parentheses operator whose return type is `const double` and which also promises not to modify any of the class' data members.

(f) Matrix addition is defined by component-wise addition; that is, the sum of two matrices `A+B` is accomplished by adding `A(i,j)+B(i,j)`. Write a method for the class that overloads the addition operator to perform matrix addition.

(g) Modify the class to a template class for matrices of type `T`.

(h) Write a driver that tests all the features of the `Matrix` class.

5. Consider the following code:

```
#include <iostream>
void gutz(char* s) {
    int i = 0, j = 0;
    for (; *s; ++s, ++i) {
        if (   tolower(*s)=='a' || tolower(*s)=='e'
            || tolower(*s)=='i' || tolower(*s)=='o'
            || tolower(*s)=='u') {
            j = 0;
            while(*s) {
                *s = *(s+1);
                ++j; ++s;
            }
            s -= j+1;
            --i;
        }
    }
    s -= i;
}

void print_array(char* s) {
    // write code here
    }
}

void test(std::string s) {
    gutz(&s[0]);
    print_array(&s[0]);
    std::cout << "\n";
}

int main() {
    std::string s;
    while (std::cin>>s && s!="quit")
        test(s);
}
```

(a) Give a detailed description of what `gutz()` does and how it works. What type of input does it take? How does it process that input?

(b) In `test()` the argument passed to `gutz()` is `&s[0]`. What is this? Could the argument just as well have been simply `s`?

(c) Write code for the body of `print_array()` that uses pointer arithmetic to print out the c-string passed to it.

6. The following recursive function prints the binary representation of a nonnegative integer:

```
void printBinary(int n) {
    if(n>1) printBinary(n/2);
    std::cout << n%2;
}
```

(a) What is the base case here?

(b) Generalize this function into a recursive function `changeBase()` that accepts two integer parameters '`n`' and '`base`' and returns '`n`' relative to a '`base`' between 2 and 10;

7. (Coding exercise) Use a laboratory computer to write code meeting the following specifications.
Of interest to mathematicians are 3-tuples of positive integers, (a, b, c) such that $c^2 = a^2 + b^2$. In the starter code below, we define a `struct Trip` to hold integers a, b, c together with a comparison operator ¡ for sorting. The program you'll develop here will be based on the idea that for any two relatively prime integers, $n < m$, that differ by an odd number $a = m^2 - n^2, b = 2mn, c = m^2 + n^2$ form a relatively unique `Triple`.
Given a maximum value for c (say, 1000) the program will write the `Triples` to a text file, something like this:

```
a: 3 b: 4 c: 5
a: 5 b: 12 c: 13
a: 15 b: 8 c: 17
...
a: 129 b: 920 c: 929
```

```
#include <vector>    // std::vector
#include <algorithm> // std::sort
#include <iostream>  // std::ostream, std::cout, std::endl
#include <fstream>
// Structure to hold triples
struct Trip {
  int a, b, c;
  bool operator< (Trip& t) { return c < t.c; } // Comparison operator for std::sort()
};
// Function to print triples
std::ostream& operator << (std::ostream& os, Trip& t) {
  //code here to output text such as "a: 3  b: 4 c: 5" and return the right sort of
    thing.
}
// Helper function to check if two integers are relatively prime
bool isRelPrime(int m, int n) {
  //for all ints from 2 to n/2
    // if m and n can be divided evenly by the same number, they are not relatively
    prime
      return false;
  return true;
}
void createTriples(int max, std::vector<Trip>& vT) {
  // for m between 2 and max
    // for n between 2 and m (n is the smaller one)
       // if m and n are relatively prime and difference is odd, we have a Trip
          // use the given formulae to create a Trip
          // When C is larger than max, return

         // otherwise, push new Trip onto the vector of Trips
      }
    }
  }
}
int main() {
  // Make a vector container to hold all triples
  // Fill the container with all triples
  // Sort the triples with sort(v.begin(),v.end())
  // Print the triples
  // Create of output file stream for "Triples.txt"
  for (std::vector<Trip>::iterator it = Triples.begin(); it != Triples.end(); ++it)
    // write the Trip to the file
}
```