Exam I (112pts)

Debugging: Find and correct each bug:(10pts) Rewrite correct code here (you may pretend you are already inside main): (2pts each)

1.
```
#include <iostream>
using namespace std;

int main(){
  std::cout << “Hi!\n”;
  return ‘0’;
}
```

2.
```
#include <iostream>

int main(){
  std::cout >> “Hi!\n”;
  return 0;
}
```

3.
```
#include <iostream>
using namespace std;

int main(){
  string c = ‘Awesome!\n’;
  cout << c;

  return 0;
}
```

4.
```
#include <iostream>
using namespace std;

int main(){
  x = “Hi!\n”;
  cout << x;

  return 0;
}
```

5.
```
#include <iostream>
using namespace std;

int main(){
  int x = 3;

  if x < 10:
     cout << “x is tiny”;

  return 0;
}
```

Use Existing Code (10pts): Using only the functions `plotX(), plotO(), and print()`, write a main function that prints this output to the console:

```
X.X
.O.
O.X
```

The variable `board` has been supplied for you and has been initialized to the string "……...". `board` keeps track of moves on a tic-tac-toe board by replacing a . with an X or an O. The first position in the string represents the upper-left hand corner of the board while the last position in the string represents the bottom right-hand corner of the board. The dots in between map to the board from left-to-right and top-to-bottom. Reading the code can help you confirm this description. Using this knowledge, place X's and O's in the string to match the above output and print your result to the console.

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


void plotX(int,int,string&);
void plotO(int,int,string&);

void print(string);

int main(){
  string board = ".........";









  return 0;
}

void plotX(int x, int y, string &board){
  board[y * 3 + x] = 'X';
}

void plotO(int x, int y, string &board){
  int index = y * 3 + x;
  board[y * 3 + x] = 'O';
}

void print(string board){
  for(int i = 0; i < board.length(); i++){
    cout << board[i];

    if((i + 1) % 3 == 0){
      cout << endl;
    }
  }
}
```

Answer the questions below: (20pts – 4pts each)

1. Which of the following if statements will print out the string “cat”

a.
```
if(true || false){
  cout << “cat”;
}
```

b.
```
if(false || false){
  cout << “cat”;
}
```

c.
```
if(true || true){
  cout << “cat”;
}
```

d.
```
if(false || true){
  cout << “cat”;
}
```

2. While a program is running, what physical component of the computer is storing the program’s variables? What is the format of the information stored there?

3. What is the difference between an if statement and a while loop?

4. When you follow the steps to perform long division by hand, is this an example of an algorithm in action, why or why not?

5. How is a human-readable C++ source file (.cpp) converted into a computer program that the CPU can run directly? Describe the process in as much detail as you can.

Trace the programs: (32pts) Please write the output of the following program in the box to the left, remember to keep track of your variable states using registers to aid your concentration.

1.

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

const int MAX = 3;

void d(string);
void g(string&, int);
int inner(int,int);
int test(int,int);

int main(){

  string s = "0";
  int m = 2;

  cout << "entering main loop..." << endl;
  for(int i = 0; i < MAX; i++){
    cout << "calling d..." << endl;
    d(s);

    cout << "calling g..." << endl;
    g(s, m);
  }
  cout << “m: “ << m << endl;

  int x = test(3, 5);
  cout << x << endl;

  if(MAX == 3){
    cout << "inside if" << endl;
  } else {
    cout << "inside else" << endl;
  }

  return 0;
}


void d(string s){
  cout << "inside d..." << endl << endl;

  for(int i = 0; i < s.length(); i++){
    cout << s << endl;
  }

  cout << endl << endl << "leaving d..." << endl;
}

void g(string &s, int m){
  cout << "inside g..." << endl;
  cout << "i: ";

  for(int i = 1; i < m; i++){
    s += s;
    cout << i;
  }

  m = m + m;
  cout << endl << "leaving g..." << endl;
}

int inner(int s, int t){
  cout << "inside inner " << endl;
  return t - s;
}

int test(int a, int b){
  cout << "inside test" << endl;
  return inner(a * 2, b * 3);
}
```

Write the program requested below: (40pts)

Write a computer program that takes in two single-character Roman numerals as input and produces their sum written in decimal. For instance, if the user enters the values V and C, the program will output 105, which is equivalent to V + C (5 + 100). Use the following table to refresh your memory for the value of each Roman numeral. Your program is only responsible for converting the numerals listed in the table below and only needs to accept a single character for each input (so for instance, the Roman numeral I is input the program should account for, but III and IV are not). Please raise your hand if you have any questions.

I	1
V	5
X	10
L	50
C	100
D	500
M	1000

Your program should consist of two functions:

`convertRoman()` - will take a single character representing one of the Roman numerals in the table to the left as input and will return the integer that corresponds to that numeral

`romanAdd()` - will take two characters representing Roman numerals in the table to the left as input and return an integer representing the sum of those two numbers. romanAdd() should call convertRoman() within itself to convert the characters it receives to integers before attempting to add them together

In main(), your program should welcome the user to the program. This should happen only once for the life of the program. The program should then ask the user for the first single character Roman numeral. If the user types ‘q’, the program should end. The program should next ask the user for the second single character Roman numeral. If the user types ‘q’, the program should end. From here, the sum should be computed and output to the console. The program will then repeat asking for two Roman numerals and adding them just as happened originally. This process should happen over and over again until the user types ‘q’ for one of the Roman numeral inputs.

Extra Space: