NAME _____

Final Exam, 3/22/2017
CIS330, Winter 2017

You have 2 hours to complete this exam. You are not allowed to use any books, notes, calculators, or electronic devices. Feel free to separate pages, and make sure to write your name on each sheet. Write your answers carefully and legibly. Partial answers are better than no answer. Feel free to skip around and go back to an earlier question later. When you are asked to write a program, writing comments is optional but you are expected to use good style and indent appropriately to make your code easier to follow. Write down assumptions you are making.

NAME: _____

1.) [10] Consider the following **C program**, stored in a file **`prog.c`** on a Unix system.

```
#include <stdio.h>

int func(int *ptr){
   *ptr = 3;
   return *ptr;
}

int main() {
   int z = 1;
   int y = func(&z);
   printf("%d\n", y);
   return 0;
}
```

(a) [2 points] Assume you are in the bash shell. How would you compile this program in bash using **gcc**?

$ gcc -std=c99 -g -o prog.exe porg.c

Acceptable minimal answer: gcc prog.c

(b) [5 points] Will the compilation succeed? If not, why? If it compiles successfully and you run the resulting executable (show the command you would use to run it below), what is the output?

Yes, code compiles.

$ ./prog.exe (or ./a.out if you didn't specify an executable name in part a)
Output:
3

(c) [3 points] What will happen if the line **"`#define z 6`"** (without the quotes) is added right after the **"`#include <stdio.h>`"** line?

Compile-time error because int z = 1; becomes

NAME: _____

2.) [10] Answer the following questions about the **C language** briefly (a sentence or two at most).

(a) [2 points] How are function arguments passed in C?

By value

(b) [2 points] How do you declare a statically allocated two-dimensional array in **C** that can hold a 10 by 5 matrix of floating-point values?

```
float var[10][5];
// or
double var[10][5];
```

(c) [2 points] Will the following **C** example compile and link successfully with the "`gcc foo.c`" command to produce an executable? If not, why not?

```
/* file: foo.c */
int doSomething() { return 1; }
```

It will not compile because there is no main function.

(d) [4 points] In **C**, what does the * operator do when used in front of a variable, e.g., in an expression such as **"`3 + *x`"**? Give a simple code fragment example to illustrate your answer (make sure to declare all variables in your example).

It dereferences a pointer, returning the value at the memory address stored in the pointer variable.

```
int a = 5;
int *x = &a;            // x stores the address of a
Int b = 2 * (*x);       // b is 10
```

NAME: _____

3.) [13 points] Write a complete **C program** that accepts a single sentence (terminated with one of .!?) as input and outputs the number of words in the sentence. You can assume that words are separated by single spaces. Structure the code into two or more functions (counting main), i.e., do not put the entire implementation inside main().

```
$ ./myprog
Please enter a sentence: I wish it would stop raining!
Number of words: 6
```

You can assume that the user input will not exceed 200 characters. Do not worry about including standard headers if you don't remember them exactly (no points deducted for missing includes).

NAME: _____

4.) [20 points] **Using C++**, define a class `WordStats` that can be used to collect the sentence statistics described below. In your main(), prompt the user to input **a paragraph consisting of one or more sentences** (each terminated with one of .?!), then create a `WordStats` object and use it to **compute and output** the following statistics:

- number of words,
- average word length
- average number of words per sentence

Do **not** use the C functions scanf, getline, strcpy, malloc, or printf. If you don't remember header names, you can omit the includes. You can assume that words are separated with spaces and the only punctuation occurs at the end of each sentence. Do not worry about including standard headers if you don't remember them exactly (no points deducted for missing includes).

```
#include <iostream>
#include <istream>

using namespace std;
class WordStats {
public:
  void readParagraph() {
    string w = "";
    string punct = ".!?";
    int wordChars = 0, wordCount = 0, sentenceCount = 0;
    while (cin >> w) { // read a word
      wordChars +=  w.length(); wordCount++;
      if (punct.find(w[w.length()-1]) ] != string::npos) {
        sentenceCount++;
      }
    }
    this->numWords = wordCount;
    this->avgWordLen = wordChars / wordCount;
    this->avgNumWordsPerSentence = wordCount / sentenceCount;
  }
  int getNumWords() { return this->numWords; }
  double getAvgWordLen() { return this->avgWordLen; }
  double getAvgWordsPerSentence() { return this->avgNumWordsPerSentence; }

private:
  int numWords;
  double avgWordLen;
  double avgNumWordsPerSentence;
};
```

```
int main(int argc, char *argvp[]) {
  WordStats ws;
  ws.readParagraph();
  cout << "Number of words: " << ws.getNumWords() << endl;
  cout << "Average word length: " << ws.getAvgWordLen() << endl;
  cout << "Average words per sentence: " << ws.getAvgWordsPerSentence()
       << endl;
  return 0;
}
```

NAME: _____

5.) [7 points] These questions are about the **C++ language.**

(a) [4 points] What is the output of the following code?

```
#include <iostream>
#include <utility>    // std::pair

class MyPair {
public:
    MyPair(std::string s, int i) { p.first = s; p.second = i; }

    friend std::ostream& operator<<(std::ostream& os, const MyPair& obj) {
      os << obj.p.second << obj.p.first << "!";
      return os;
    }

    std::pair<std::string,int> p;
};

int main() {
    std::cout << MyPair("potatoes",3) << std::endl;
}
```

3potatoes!

(b) [3 points] What are **three C++ operators** that cannot be overloaded?

.
::
? :
Unary * (dereferencing)
.*

NAME: _____

6.) [10] Specify the **Unix shell (bash) command(s)** for each task below (multiple correct answers are possible, just give one for each part).

(a) [3 points] List *all* files in the current directory, displaying the files' permissions.

ls -al

(b) [1 point] Change the current directory to `/some/dir` .

cd /some/dir

(c) [1 points] Delete `myprog.c` in the current directory.

rm myprog.c

(d) [1 point] Create a subdirectory `newdir/` in the current directory.

mkdir newdir

(e) [1 point] List/view processes running on the machine.

ps -a
ps -fe

(f) [1 point] Kill a process with PID 123.

kill 123

(g) [2 points] Find a file named `foo.c` in any subdirectory starting from the current directory.

find . -name foo.c

NAME: _____

7.) [10] Consider a C++ class **`PairOfDice`** that stores the values on two N-sided dice. The constructor should create two random die values, e.g., the call

```
PairOfDice myDice(8);
```

would create two dice with values [1,8] (inclusive).

```
// File: dice.hpp
class PairOfDice {
public:
     PairOfDice(int sides);
     // … more functions …
private:
     // data members:
     int sides;
     int die1;
     int die2;
};
```

(a) [5] Add some data members to the above class and then write the definition of the constructor as it would appear in the file `pairofdice.cpp`. Extra credit [+1 pt]: write an additional copy constructor.

```
//Assume srand(time(NULL)); is called in the beginning of main
#include<cstdlib>
#include<ctime>
PairOfDice::PairOfDice(int s) : sides(s) {
      die1=(rand()%s)+1;
      die2=(rand()%s)+1; }
}

// Extra:
PairOfDice::PairOfDice(const PairOfDice& orig) {
      die1=orig.die1;
      die2=orig.die2;
}
```

(b) [5] We wish to compare two `PairOfDice` objects based on the sum of current values of both dice. For example, the dice {3,1} would be less than the dice {5,1} because 4 is less than 6. Write a member function to compare two PairOfDice objects based on the sum of their values. (Extra credit [+2]: Write the definition of a member function that overloads the < comparison as used below: `if (myDice < yourDice)`).

```
bool PairOfDice::operator<(const PairOfDice& other) {
      return (this->die1 + this->die2) < (other.die1 + other.die2);
}
```

NAME: _____

8.) [23 points] Implement the *matching pennies* game with the following rules. The game is played between two players, Player A (human) and Player B (computer). Each player has a penny and must secretly turn the penny to heads or tails. The players then reveal their choices simultaneously. If the pennies match (both heads or both tails) Player A keeps both pennies, so wins one from Player B (+1 for A, -1 for B). If the pennies do not match (one heads and one tails) Player B keeps both pennies, so receives one from Player A (-1 for A, +1 for B). Assume each player starts with 5 pennies.

You can use either C or C++. In your implementation, the computer randomly chooses heads or tails. The user inputs either "heads" or "tails" (after flipping a coin). The game continues until the user inputs "quit" or when either player runs out of pennies. Before exiting, the program prints the total number of pennies for both players. Here is an example game session:

```
Computer has 5 pennies, and you have 5 pennies.
Heads or tails (quit to exit)? heads
Computer picked tails. Computer wins this round. [You: 4, Computer: 6]
Heads or tails (quit to exit)? heads
Computer picked heads. You win this round. [You: 5, Computer: 5]
Heads or tails (quit to exit)? tails
Computer picked tails. You win this round. [You: 6, Computer: 4]
Heads or tails (quit to exit)? quit
Game over. You have 6 pennies, and the computer has 4 pennies. You won!
```

Do not worry about including standard headers if you don't remember them exactly (no points deducted for missing includes).