PIC 10A
Introduction to Programming

Midterm

Instructions...

- Gradescope...
  - You have until **Friday May 7** at 11:59pm PST to submit your solutions to Gradescope.
  - Make sure that you correctly tell Gradescope on which **pages you answer each question**.
- Class on Monday May 3 at 1pm is cancelled to give you more time this week.
- Exam conditions???
  - I think it is beneficial for you to attempt the exam under exam conditions **at first**.
  - After you have evaluated your progress by attempting the exam under exam conditions, you can spend as long as you wish writing up **perfect solutions** with full explanations.
  - **Only** submit your perfect solutions. We do **not** need to see your previous attempts.

Name: _____
Student ID number: _____
Discussion: _____

Question	Points	Score
1	6	
2	6	
3	6	
Total:	18	

In **every** question, you should assume that

```
#include <iostream>
#include <string>

using namespace std;
```

has been typed at the start.

Problem 1. *6pts.*

Explain the output of the following code.

```
int main() {
    cout << boolalpha;
    cout << (100.0 * 20.15 < 2015.0) << endl;
    cout << static_cast<int>(100.0 * 20.15) << endl;

    cout << 'g' - 'b' << endl;
    char ch = 'D' + 'a' - 'A';
    cout << ch << endl;

    string s = "AAARGH!!!";
    if (s.find("AAA")) { cout << 1 << endl; }
    if (s.find("RGH")) { cout << 2 << endl; }
    if (s.find("???")) { cout << 3 << endl; }

    return 0;
}
```

For full credit, your explanation **must** use the following words **appropriately**:

- `int`, `double`, `bool`, `char`, `size_t`,
- `static_cast<int>`, `static_cast<bool>`, `static_cast<char>`, `static_cast<size_t>`,
- console, display / displays / displayed, assign / assigns / assigned / assignment,
- implicit, convert / converts / converted / conversion, zero, non-zero,
- rounding, truncate / truncates / truncated / truncation.

Problem 2. *6pts.*

Explain the output of the following code by...

- carefully keeping track of the input buffer (you should **clearly display** the contents of the **input buffer** after **every** significant line of code);
- carefully following the instructions on pages 5 and 6 of the supplementary materials which describe how `cin >> variable`, `getline(cin, s)`, `cin.ignore()`, `cin.get()`, `cin.peek()` work (e.g. you should **explicitly** use steps 1 to 4 for `cin >> variable`).

```
int main() {
    cout << "Type (not copy and paste) the four (not three)"           << endl;
    cout << "commented lines of code (ending each by pressing ENTER):" << endl;
/*

9 8
7 6543
2 1012 345 678 911
*/
    int    i1, i2, i3, i4, i5;
    char   c1, c2;
    string s;

    cin >> i1;
    cin >> i2;
    getline(cin, s);

    cin >> i3;
    cin.ignore();

    c1 = cin.peek();
    c2 = cin.get();

    cin >> i4 >> i5;
    cout << endl;

    cout << "Line 1: " << i1 << endl;
    cout << "Line 2: " << i2 << endl;     // These variables
    cout << "Line 3: " << s  << endl;     // are printed in
    cout << "Line 4: " << i3 << endl;     // the same order
    cout << "Line 5: " << c1 << endl;     // that they are
    cout << "Line 6: " << c2 << endl;     // assigned to.
    cout << "Line 7: " << i4 << endl;
    cout << "Line 8: " << i5 << endl;

    return 0;
}
```

Problem 3. *6pts.*

Explain the output of the following code with the aid of a **picture**.

```
int f(int& i, int j) {
    int tmp = i;
    i = j;

    if (tmp == 8) { cout << tmp << ' ' << i << ' ' << j << endl; }

    j = tmp;
    return j;
    return i;
}
int main() {
    int i = 8, j = 1, k = 0;

    f(j, k);
    cout << i << ' ' << j << ' ' << k << endl;

    i = f(i, j);
    cout << i << ' ' << j << ' ' << k << endl;

    return 0;
}
```

For full credit, your picture **must**...

- display a **function scope for each function call**;
- display all **function parameters** in the appropriate place;
- draw **references** consistently with how they were drawn in lecture;
- indicate the **full history of values** that each non-referencing variable has;
- indicate the **order** that values are updated or introduced, and scopes are introduced and destroyed.

You can satisfy the last bullet point by labelling your picture with numbers in a different color to your normal writing. I am happy to demonstrate this idea in office hours. You should expect to use at least the numbers from 1 to 11. It is reasonable to lump together the introduction of a function scope and the initialization of the function parameters. It is also reasonable to lump together the destruction of a function scope and the impact of a `return`ed value. This video (a link you can click) should be useful.

For full credit, your prose **must**...

- **disambiguate** variables in `main` and variables in a function scope (if they happen to have the same name);
- **explain** issues concerning the `return` keyword carefully.