General Instructions: This exam must be turned in at the beginning of class on the day that it is due. It may not be turned in late. All questions must be worked on and completed individually. You may use online references as necessary, but the answer that you submit must be wholly your own. You may not collaborate with other students (regardless of whether or not they are enrolled in this class). You may, however, speak to the TAs and ask questions if you need clarification on any question.

Note: Your “Net ID” is the identifier that you use when logging into the school’s servers. Your “NDID” is a 9-digit numeric identifier which was issued to you by the school, and can be found on your Student ID Card.

Question 1.
What is your NDID? ________________________________

Convert it to hexadecimal. ________________________________

What is the minimum number of bits required to represent your personal NDID in binary (assume that it is an unsigned number)? ________________________________

What is the minimum number of bytes required to represent your personal NDID in binary (assume that it is an unsigned number)? ________________________________

Question 2.
We use both cout and cerr in our programs, but they both seem to just print to the command line when they run. In what ways do they behave differently? Give an example of why this might be useful.
**Question 3.**

What is the *correct* output of the following program? Explain your answer.

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

int main() {
    int n = 3;
    n += ++n + n++ + --n;
    cout << n << endl;
}
```

**Question 4.**

Explain each of the following snippets and describe what they are doing. Specifically, identify the unique behaviors of *a* as compared to *b*, and *c* as compared to *d*.

```cpp
char ch;
cin >> ch;
```
```cpp
char ch;
cin.get(ch);
```
```cpp
char ch[50];
cin >> ch;
```
```cpp
char ch[50];
cin.getline(ch, 50);
```
Question 5.
The following code compiles and runs, but sometimes it will crash, producing a segmentation fault. What is wrong with it and why does it sometimes produce the error?

```cpp
#include <iostream>
using namespace std;
int main() {
    char a[50];
    while (cin >> a) {
        cout << "Memory location of a is " << &a << endl;
        cout << a << endl;
    }
}
```

Question 6.
Assume that the variable `array` is initialized as a `char*` that is pointing to a valid C-string. What is the difference between the following two expressions: `(*array)++` and `*(array++)`.

Consider this expression: `*array++`. Is this equivalent to `(*array)++` or `*(array++)`? Will your answer be the same if you use a different compiler? Justify your answer.
Question 7.
The following code was written by a programmer who does not fully understand the purpose of `const`, and as a result his code doesn't compile. Of the 6 `const` expressions that appear in the code, identify which one(s) are correct and should be there, which one(s) should not be there, and which one(s) may be omitted because they are superfluous. Explain your reasoning. Remember that sometimes `const` is required for technical reasons, and sometimes it is used for philosophical reasons (e.g. it communicates additional information about the code). *Hint:* Just because the code compiles does not mean that the answer is “correct”. Think about what the `const` keyword communicates about the code. If the variable represents something that should not be modified, then it should be declared `const`. There is (in no particular order), three of one type, two of another type, and one of another type.

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

void populate(const int * array, const int total) {
    const int * end = array + total;
    while (array != end)
        cin >> *array++;
}

void print(const int * array2, const int total2) {
    const int * end2 = array2 + total2;
    while (array2 != end2)
        cout << *array2++ << endl;
}

int main() {
    int number;
    cin >> number;
    int array[number];
    populate(array, number);
    print(array, number);
}
```
Question 8.
Examine the following two programs. They are identical except for one line, and yet they produce surprisingly different output. Answer the following questions: 1.) Which program seems to operate correctly, and which one seems to have a bug? 2.) Explain why the programs operate differently. Give as much detail as possible about how this difference is manifested in this program so as to prove that you understand the complexity of the problem. 3.) What is it about the output makes this “bug” so difficult to find?

Program 1
1 #include <iostream>
2 #include <iomanip>
3 #define NUMBER float
4 using namespace std;
5 int main() {
6 NUMBER accumulator = 0;
7 int count = 0;
8 cout << setprecision(2);
9 while (accumulator < 1) {
10 cout << count << "\t"
11 << accumulator << endl;
12 accumulator += .01;
13 count++;
14 }
15 }

Program 2
1 #include <iostream>
2 #include <iomanip>
3 #define NUMBER double
4 using namespace std;
5 int main() {
6 NUMBER accumulator = 0;
7 int count = 0;
8 cout << setprecision(2);
9 while (accumulator < 1) {
10 cout << count << "\t"
11 << accumulator << endl;
12 accumulator += .01;
13 count++;
14 }
15 }
Question 9.
Compile this program on student00.cse.nd.edu. What is the last line that is printed by the following program? Explain what is happening.

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

int main() {
    char a[] = "0123456789ABCDE";
    char b[] = ".................";
    b[strlen(b)] = '-';
    cout << &a << endl;
    cout << &b << endl;
    cout << a - b << endl;
    cout << &b[14] << endl;
}
```