Name:

Score __105___ / 100

1. (10 points) The following code is part of a linked list class and is, technically, a perfectly legal use of overloaded operators as far as the compiler is concerned. Why shouldn't it be used in practice?

```
void LinkedList::operator[] (int num) {
      // remove the first num entries from the list
      for(int count = 0; count < num && head != 0; count++) {
            Node * curr = head;
            head = head->next;
            delete curr;
      }
}
```

The [] operator is normally used to access ordered elements in a data structure. Using it to delete elements is breaks this convention and is non-obvious.

2. (15 points) A developer is creating a C++ library for reading and writing XML files. There are a number of errors that can crop up in the course of reading an XML file (files that fail to open, improperly formatted files, etc) and the library needs to be able to deal with those errors. Assuming programs using the library should be able to recover from errors in the library if they wish, should the developer use thrown exceptions or assertions for error handling in the library? Why?

The library should throw exceptions, as we need to given the programs using the library the chance to recover from errors.

3. (10 points) A C++ program under development relies heavily on dynamically allocated memory. During testing, it is revealed that the program is suffering from frequent segmentation faults. Trace debugging does not reveal anything useful about why the seg faults are occurring. What should be done next to debug the program? (No, “ask the developer's C++ instructor” is not an acceptable answer. Sorry.)

Use Valgrind to track down the cause of the segmentation faults.

4. (15 points) What output would be written to "output.txt" by the following code segment, assuming the file "input.txt" contains "3 foo bar baz"? (For convenience, the code below uses a "␣" to indicate spaces in strings. Do the same in your output.)

```
#include <string>
#include <fstream>
#include <iomanip>

using namespace std;

int main() {
    int count;
    ifstream in ("input.txt");
    ofstream out ("output.txt");
    in >> count;
    string str[count];
    for(int i = 0; i < count; i++) {
        in >> str[i];
    }
    out << "|";
    for (int i = 0; i < count; i++) {
        out << setw(str[i].length()) << i << "␣␣|";
    }
    out << endl;
    out << "|␣";
    for (int i = 0; i < count; i++) {
        out << setw(str[i].length()) << str[i] << "␣|␣";
    }
    out << endl;
    return (0);
}
```

|␣␣0␣␣|␣␣1␣␣|␣␣2␣␣|
|␣foo␣|␣bar␣|␣baz␣|␣

5. (15 points) Describe what the following function does, being sure to mention details related to the types of the arguments. Also identify any special requirements placed on the types of the arguments.

```
template <typename T, typename U>
T GetMin (T a, U b) {
      return (a<b?a:b);
}
```

The function receives two values of potentially different types and returns the smaller value cast to the type of the first argument. It must be possible to compare the types using <, and to cast type U to type T.

6. (10 points) What general purpose is served by an STL iterator?

An iterator provides a way to move through a data structure and access its contents without needing to know or understand the details of the data structure's implementation.

7. (5 *bonus* points) What is the air speed velocity of an unladen swallow?

Any answer gets four points. Answers of the form "is it an African or a European swallow?" get all five.

8. (15 points) Below is a simple C++ program. Describe what would need to be done to convert it to a C program. You DO NOT need to actually write the C version. (Hint: there are three major types of changes to be made.)

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

int main() {
      cout << "how many values? ";
      int num;
      cin >> num;
      int * arr = new int[num];
      for(int i = 0; i < num; i++) {
            cin >> arr[i];
      }
      delete arr;
}
```

1) Switch from iostream to stdio (or, equivalently, use printf/scanf instead of cout/cin)
2) Move all variable declarations to the beginning of the function
3) Use malloc()/free() instead of new/delete

9. (10 points) Briefly describe the differences between statically and dynamically linked libraries.

Statically linked libraries are compiled into the executable, making the executable bigger but preventing problems with missing or incompatible libraries. Dynamically linked libraries are loaded from a standard location at runtime, reducing executable sizes and making it easier to update libraries, but potentially introducing compatibility problems.