CS 7B - Fall 2019 - Quiz 10/16 —Name:

---

1. Consider the following complete program:

```
#include "std_lib_facilities.h"
void doThis(ifstream& ifs);
void doThat(ifstream& ifs);
int main() {
  cout << "Enter the name of a file to convert to lower case: ";
  string fname;   cin >> fname;
  ifstream fs(fname);
  doThis(fs);
  fs.clear();   fs.seekg(0, std::ios::beg);
  doThat(fs);
}
void doThat(ifstream& ifs) {
  cout << "Enter output file name: ";
  string output;
  cin >> output;
  ofstream ofs(output.c_str());
  if (!ofs) error("can't open output file ", output);
  char ch;
  while (ifs.get(ch)) {
    if (isalpha(ch)) ch = tolower(ch);
    ofs << ch;
  }
}
void doThis(ifstream& ifs) {
  char c{ 0 };
  ifs >> c;
  char* buffer = &c;
  ifs.unget();
  ofstream ofs("outputFile.txt");
  char chin{ 0 };
  while (ifs.read(buffer, 1)) {
    if (tolower(*buffer) == 32) ofs << ' ';
    ofs << char(tolower(*buffer));
  }
}
```

(a) What type of variable is `buffer` in `doThis()`? How is it initialized?

ANS: The variable, `buffer` is declared as a `char` pointer on line 27. It's initialized as the address of `c` which was declared as a NUL character (not a null pointer, mind you.) If you write char c = '\0', it's the same as char c = 0, the character used to terminate a string. But then we extracted the first `char` from the input file stream and took the address of that to initialize `buffer`.

(b) What is happening on line 28? Why is this done?

ANS: When `ifs.unget()` is called, the current location in the stream is decreased by one character, making the last character extracted from the stream once again available to be extracted by input operations.

This is done so we can execute the `while` loop from the beginning.

(c) What is the meaning of the condition for the `while` loop on line 31? How can it turn out to be `false`?// **ANS:**

ANS: The `read()` function of the `iostream` library takes to parameters, a `char*` to indicate the memory location to start reading data, and an `int` to indicated the size of the block to be read from the input stream. The `read()` function returns the `istream` object (`*this`) which, if it's `eofbit`, `failbit` or `badbit` flags have not been set, will be interpretted as `true`. What we expect to happen is that it will be `false` when the `eofbit` is set by reaching the end of the input stream (end of file.)

(d) What is the meaning of the conditional on line 32? How can it turn out to be `true`?

ANS: The comparison "`tolower(*buffer) == 32`" dereferences the `char* buffer`, applies the `tolower()` function to that `char` value (which will do nothing since ascii value 64 is not an alpha character) and then compares that ascii value with 32, the ascii value for the space character...so it would only be true if `*buffer` is the space character.

(e) Compare that `while` loop of `doThat()` with the `while` loop of `doThis()`. How is it different? How is it the same?

ANS: The two loops are the same in that the precondition/postcondition (input/output) specifications are the same. The `doThat() while` loop is perhaps simpler and easier to read in that it doesn't require a `char*` declaration and uses the `get(ch)` function in the `istream` library and checks to see if it's an alpha character before calling `tolower()`.