### Grades

<table>
<thead>
<tr>
<th>Score</th>
<th>Num Students</th>
<th>Approx Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>90s</td>
<td>7</td>
<td>A</td>
</tr>
<tr>
<td>80s</td>
<td>20</td>
<td>B</td>
</tr>
<tr>
<td>70s</td>
<td>9</td>
<td>C</td>
</tr>
<tr>
<td>60s</td>
<td>9</td>
<td>C</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Score</td>
<td>Num Students</td>
<td>Approx Grade</td>
</tr>
<tr>
<td>-------</td>
<td>--------------</td>
<td>--------------</td>
</tr>
<tr>
<td>50s</td>
<td>7</td>
<td>C</td>
</tr>
<tr>
<td>40s</td>
<td>5</td>
<td>D</td>
</tr>
<tr>
<td>30s</td>
<td>3</td>
<td>D</td>
</tr>
<tr>
<td>20s</td>
<td>1</td>
<td>F</td>
</tr>
</tbody>
</table>

### Notes (by question)

1. Use `exit` to avoid long if-then-else constructs. This is a standard idiom for handling files.

2. Read the question. You are to write a function. Your function operates on a character array. The function should contain no stream IO! The example only shows how your function might be used.

   Make sure you read the prototype of a function. Note that the prototype of `toupper` is `char toupper (char)` and not `void toupper (&char)`. The prototype tells you how to use the function. The call must be `s[i] = toupper(s[i]);` and not `toupper(s[i]);`

   It is better to avoid nested loops when they are not necessary, as here. It easy to mess up the termination conditions. The following code may lead to a segmentation fault (why?):

   ```c
   while (s[i] != '\0') {
       if (s[i] == '#') {
           while (s[i] != '\0') {
               if (islower(s[i])) s[i] = toupper(s[i]);
               i++;
           }
       }
       i++;
   }
   ```

3. In order to reverse a string, you do not need to know it’s length in advance (you can use `strlen`). In order to reverse an array, you do need to know the length in advance.

   Arguments that are passed by value do not to be declared as `const` (although it is not an error). Arguments pass by reference (such as arrays) may be usefully declared `const` in many cases, like the last question here.

4. To get credit you had to show the intermediate steps of horner’s rule.

5. The second question would be more meaningful if I had used `const int& f (const int a[])`. Many of your answers were overly vague: “the function takes a constant integer array”. What does that mean?

   The third question was worth one point. It is discussed in the text, although I did not discuss this in lecture. About one third of you got it right.

6. I did not take off points if you missed the arch from node 1 to node 3. The other edges were worth about 2pts each. 2pts for indicating output correctly.

7. The last of the four questions was subtle. I accepted many answers.

8. Note that data members do not need to be passed as arguments to member functions. That is one of the great things about C++!
Command Line Arguments, and File I/O

1. [15 points] In this problem you will write a program (called copy) that copies the contents of one file to another. The file names will be provided as command line arguments. For example, the command

   `copy old.txt new.txt`

should copy the contents of `old.txt` to the file `new.txt`. If the directory already contains a file named `new.txt`, it is overwritten.

You are to define a program (the contents of `copy.cpp`) that implements this behavior. The program expects exactly two arguments. If the number of arguments is not exactly two, the program should print an error message and exit. Otherwise, the files should be opened and the contents copied from the first to the second. If one of the files cannot be opened, an error message should be printed. You need not `#include` any header files.

ANSWER:

```c++
#include <fstream.h>

main (int argc, char **argv)
{
    if (argc != 3) {
        cerr << "usage: " << argv[0] << " infile outfile" << endl;
        exit(1);
    }

    ifstream in(argv[1]);
    if (! in) {
        cerr << argv[0] << ": unable to open " << argv[1] << " for input" << endl;
        exit(1);
    }

    ofstream out(argv[2]);
    if (! out) {
        cerr << argv[0] << ": unable to open " << argv[2] << " for input" << endl;
        exit(1);
    }

    char c;
    while (in.get(c)) // OR
        out << c; // out.put(c);

    in.close();
    out.close();
}
```

Score on this page: 2
Strings and Arrays

2. [10 points] Write a function `up` that takes a string and converts all lowercase characters after the character # (if it occurs) to uppercase, in place. For example, the fragment

```c
char s[50] = "Fox #3 is (Happy)";
up(s);
cout << s << endl;
```

should print \textbf{Fox #3 IS (HAPPY)}. You need not \texttt{#include} anything. You may use the following functions.

- `char toupper (char c);`
- `bool isupper (char c);`
- `char tolower (char c);`
- `bool islower (char c);`

**ANSWER:**

```c
void up(char s[]) {
    int i(0);
    while (s[i] != '#' && s[i] != '\0')
        i++;
    while (s[i] != '\0') {
        if (islower(s[i])) s[i] = toupper(s[i]);
        i++;
    }
}
```

3. [15 points] Give the \textit{prototype} of the following functions. Your prototype must include the minimal necessary arguments and should be as restrictive as possible (use \texttt{const} when possible).

- A function `sreverse` that reverses a string in place.
  ```c
  void sreverse (char s[]);
  ```

- A function `ireverse` that reverses an integer array in place.
  ```c
  void ireverse (int a[], int length);
  ```

- A function `imember` that determines whether an integer is an element of an integer array, returning true or false.
  ```c
  bool imember (const int a[], int length, int element);
  ```

Score on this page_______
Horner’s Rule

4. [5 points] Show all intermediate steps of using Horner’s rule to convert the string of bits (binary “digits”) 10011 to an integer. Do not write any C++ code — just compute the correct integer and show all the steps.

\[
\begin{array}{cccccc}
0 & 2 & 4 & 9 & 18 \\
1 & 0 & 0 & 1 & 1 \\
1 & 2 & 4 & 8 & 19 \\
\end{array}
\]

Classes (part I)

5. [12 points] Answer the following:

- Name three ways that constructors are distinguished from other member functions.

  - Name is same as class
  - No return type given in prototype
  - Automatically invoked when class instance created
  - Cannot be invoked on existing class instance
  - A default constructor is defined, if none defined explicitly
  - Can contain an initialization list

- Consider the member function prototype \texttt{const int f(const int[]) const;}. What are the meanings of the three instances of the keyword \texttt{const}? (Identify them as first, second, third.)

  - \textbf{first}: user cannot alter return value (in this case it changes nothing, since return is \textit{by value}, rather than by reference)
  - \textbf{second}: function cannot modify array argument
  - \textbf{third}: function cannot modify data members

- Both \texttt{struct}s and \texttt{class}es can have member functions and data members. What difference is there between them in the language definition?

  By default, \texttt{struct} members are public, \texttt{class} members are private.

Score on this page______
Finite-State Machines

6. [20 points] Background: Just to refresh your memory, the following finite-state machine (FSM) echoes the A’s in a text stream (and does not echo the rest).

```
<table>
<thead>
<tr>
<th>Start</th>
<th>A :: echo</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>AGREEABLE</td>
<td>AA</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Fabulous</td>
<td></td>
</tr>
</tbody>
</table>
```

Question: You are to draw a FSM that removes C++-style comments from a text stream. You can pretend that C-style comments do not exist (treat them like normal text). Your FSM must correctly handle strings (in matching double quotes) and strings that contain double quote characters.

Recall that a C++ comment begins with the two characters // and continues to the end of the line. A string starts with the character " and ends with the character ", as long as the final " is not preceded by a backslash. Whether the quotation or the comment takes precedence depends on which starts first.

For example, your FSM should convert the input

```cpp
/* a program // by me
 */
char* x = "hello//\"//buddy\"\"; // a comment
// The \" sequence escapes the subsequent character
main () {} // doesn’t do "much"
```

into the output:

```cpp
/* a program
 */
char* x = "hello//\"//buddy\"\";
main () {};
```

You may put your answer on the next page.

Clarification: you can assume that the character sequence \" never occurs outside a string (i.e., \" only occurs inside " . . . ").
Separate Compilation and Makefiles

7. [8 points] Consider the following makefile:

```
it :
    it.o ms.o ; g++ -o it it.o ms.o
it.h :
    ms.h
it.o :
    it.cpp it.h ; g++ -c it.cpp
ms.o :
    ms.cpp ms.h ; g++ -c ms.cpp

clean :
    ; rm *.o it
```

Suppose it is up to date. List the commands that would be executed as a result of each of the following, independently. (“touch file” has the same effect as changing file.)

- touch ms.cpp; make it
  ```
g++ -c ms.cpp
g++ -o it it.o ms.o
  ```

- make clean; make it
  ```
  rm *.o it
g++ -c it.cpp
g++ -c ms.cpp
g++ -o it it.o ms.o
  ```

- touch it.h; make it
  ```
g++ -c it.cpp
g++ -o it it.o ms.o
  ```

- touch ms.h; make it
  ```
g++ -c ms.cpp
g++ -o it it.o ms.o
  ```

Score on this page: 6
Classes (part II)

8. [15 points] Consider the following description of a car:

<table>
<thead>
<tr>
<th>Object Type: Car</th>
</tr>
</thead>
<tbody>
<tr>
<td>Knows:</td>
</tr>
<tr>
<td>The amount of gas in its tank</td>
</tr>
<tr>
<td>The capacity of its gas tank</td>
</tr>
<tr>
<td>The number of miles per gallon it can travel</td>
</tr>
<tr>
<td>Can do:</td>
</tr>
<tr>
<td>Drive a given number of miles (or until the tank is empty)</td>
</tr>
<tr>
<td>Fill the gas tank</td>
</tr>
<tr>
<td>Tell the number of miles it can travel (given current amount of gas)</td>
</tr>
</tbody>
</table>

When new cars are created, they should have a full tank.
Define a class named `Car` for this design. You need only give the declaration of the class. You need not define any member functions. Be sure to include `const` declarations where possible.

Clarification: When new cars are created, they have a full tank. The creator must specify the MPG and tank size. Use `float` or `double` for these.

ANSWER:

```cpp
class Car {
public:
    Car(float mpg, float capacity);
    // OR
    // Car(float mpg, float capacity);
    // : gas_(capacity), mpg_(mpg), capacity_(capacity)
    // {};
    void drive(float nummiles);
    void fill();
    float miles_to_empty () const;

private:
    float gas_;  
    const float mpg_; 
    const float capacity_; 
};
```

Score on this page: 7