Northern Michigan University (Marquette Co, MI)

CS201-01-22F: Introduction to C++ (Andrew A. Poe) Name: _____
Final Examination Page 1/3 Tuesday 13 December 2022 10:00 A.M. EST

Time: 110 minutes

For the following programs, write the code as directed. Do not worry about #include files or minor syntactic errors such as semicolons or matching braces. However, your code should be as correct as possible. Make sure all special cases are handled correctly. Make sure you call and use all methods correctly. If you write more than one method to solve a problem, make sure you declare your headers correctly, but you don't have to worry about prototypes.

1. Write code for **`string LiveEvil (string s);`** This method replaces all instances of "live" with "evil" in string s and returns the new string. For example, if the string were "I'm going to live life lively.", the method would return "I'm going to evil life evilly." "life" will always be lowercase; you don't have to worry about uppercase in this problem.

```
string LiveEvil (string s) {

 for (int i=0; i < s.length()-3; i++)
  if (s.substr(i,4)=="live") {
   s[i]='e';
   s[i+1]='v';
   s[i+2]='i';
   s[i+3]='l';
   i+=3;
  }
 return s;
}
```

2. Write code for **`bool FindDog (char **M, int r, int c).`**

M is a two-dimensional array of char, containing r rows and c columns. This method returns true if DOG can be found horizonally OR vertically somewhere in the array, false if it can't. (Don't worry about it appearing backwards or diagonally. Just check for its presence from left to right AND from up to down.)

Example: r=5, c=4: returns true

```
ABCD
EFDH
IJOL
MNGP
QRST
```

```
bool FindDog (char **M, int r, int c) {

 for (int i=0; i < r; i++)
  for (int j=0; j < c; j++) {
   if (j+2 < c && M[i][j]=='D' && M[i][j+1]=='O' && M[i][j+2]=='G') return
true;
   if (i+2 < r && M[i][j]=='D' && M[i+1][j]=='O' && M[i+2][j]=='G') return
true;
```

```
  }
 return false;
}
```

3. Write the code for method **`string LL::NextAfter (string s)`**. LL is a linked list defined as follows:

```
class LL {
 private:
  LL *head;
};
```

```
class LLN {
 private:
  string word;
  LLN *next;
};
```

This method returns the LOWEST string (alphabetically) in the linked list that follows string s alphabetically. For example, if the list were I-->LIKE-->TRAFFIC-->LIGHTS , NextAfter ("LIGHTS") would return "LIKE" since there are two strings that come after "LIGHTS" alphabetically (LIKE and TRAFFIC) and LIKE is the lower of the two. String s might be in the list itself, or it might not. Do not use loops; use recursion only. If there is no valid string, return "".

```
string LL::NextAfter (string s) {

 if (!head) return "";
 string l = head->NextAfter (s,head->getword());
 if (l < s) return "";
 return s;
}

string LLN::NextAfter (string s, string lsf) {

 if (word > s) && (lsf < s || word < lsf) lsf = word;
 if (next) return next->NextAfter (s,lsf);
 return lsf;
}
```

4. Write the code for method **`void LL::DelAllAfter (string s)`**. LL is a linked list defined above:

For this problem, remove from the linked list all nodes that come after string alphabetically. For example, if the list were I-->LIKE-->TRAFFIC-->LIGHTS, DelAllAfter ("LIGHTS") would result in the list becoming I-->LIGHTS. Do not use loops; use recursion only. You may assume that appropriate accessors and mutators exist, but you must write all other helper methods that you use. Do not worry about memory leaks.

```
void LL::DelAllAfter (string s) {

 if (!head) return;
 head->DelAllAfter (s,nullptr,this);
}
```

```
void LLN::DelAllAfter (string s, LLN *prev, LL *L) {

 if (next) next->DelAllAfter (s,this,L);
 if (word <= s) return;
 if (prev) prev->setnext (next);
 else L->sethead (next);
}
```

5. Write code for the method **`void LiveEvilInFile (string f)`**. This method opens the file whose name is specified in *f*. Much like Problem 1, you are to open the file whose name is specified in f. Replace all instances of "live" (lowercase) with "evil" in that file. Do not make a new file; just modify the existing one. For example, if the file were "I'm going to live life lively.", the file would become "I'm going to evil life evilly."

```
void LiveEvilInFile (string f) {

 fstream fl (f,ios::in|ios::out|ios::binary);
 fl.seekg (0,ios::end);
 int len = fl.tellg();
 for (int i=0; i < len-3;i++) {
  fl.seekg (i);
  string word = "";
  for (int j=0; j < 4; j++)
   char c;
   fl.get (c);
   word += c;
  }
  if (word=="live") {
   fl.seekp (i);
   fl << "evil";
   i += 3;
  }
 }
 fl.close ();
}
```

6. Given the following class:

```
class Stuff {

 public:
  virtual void cmp (Stuff *b) = 0;
  //a->cmp(b) > 0 if a > b
  //a->cmp(b) < 0 if a < b
  //a->cmp(b) == 0 if a == b
};
```

Write code for **`void SwapFirstLast(Stuff **A, int sz)`**.

This method finds the largest and smallest elements in array A of size sz and switches them in the array, leaving the other elements untouched. For example, if the array held ints, and the array was {4,9,8,5,5}, the array would be modified to {9,4,8,5,5}, since 4 is the smallest and 9 is the largest.

```
void SwapFirstLast (Stuff **A, int sz) {

 int lgpos = 0, smpos = 0;
 for (int i=1; i < sz; i++)
  if (A[i].cmp(A[lgpos]) > 0) lgpos = i;
  else if (A[i].cmp(A[smpos) < 0) smpos = i;
 Stuff *t = A[lgpos];
 A[lgpos] = A[smpos];
 A[smpos] = t;
}
```