PIC 10A Disc 5A Midterm Review Worksheet

Name: \_\_\_\_\_\_\_\_\_\_\_\_\_

1. Assume you’ve already written code to make two strings called haystack and needle. **Without using std::string::operator=, w**rite code that finds the last instance of needle in haystack, represented as follows: the index where the last instance begins, and the index *one after* the end of that copy of needle. (So if needle is “ab” and haystack “nabmabc”, you should say that the last instance starts at index 4 and ends just before index 6.)
*Hint: loop over the indices for haystack. For each index, loop until you find a mismatch in needle.*
int start=0, end=0;
for(int index=0; index < size(haystack) – size(needle); ++index)
{
 int jndex;
 for(jndex=0; jndex < size(needle); ++needle)
 {
 if(needle[jndex]!=haystack[index + jndex])
 {
 break;
 }
 }
 if(size(needle)==jndex)
 {
 start = index;
 end = jndex;
 }
}
2. Your solution to #1 probably involved two loops, one of which compared part of a string to needle. Now extract the body of that loop to a separate function, called strComp:
void strComp(std::string const &needle, std::string const &haystack,
 int index, int &start, int &end)
{
 int jndex;
 for(jndex=0; jndex < size(needle); ++needle)
 {
 if(needle[jndex]!=haystack[index + jndex])
 {
 return;
 }
 }
 start = index;
 end = jndex;
}
3. Define a new class:
class strView { public: int start, end; };
Make strComp operate on objects of type strView.
void strComp(std::string const &needle, std::string const &haystack,
 int index, strView &retval)
{
 int jndex;
 for(jndex=0; jndex < size(needle); ++needle)
 {
 if(needle[jndex]!=haystack[index + jndex])
 {
 return;
 }
 }
 retval = strView{index, jndex};
}
4. Now make strComp a member function of strView.
/\*In header file:\*/
class strView
{
public:
 int start, end;
 void strComp(std::string const &needle,
 std::string const &haystack, int index);
};

/\*In your code file:\*/
void strView::strComp(std::string const &needle,
 std::string const &haystack, int index)
{
 int jndex;
 for(jndex=0; jndex < size(needle); ++needle)
 {
 if(needle[jndex]!=haystack[index + jndex])
 {
 return;
 }
 }
 start = index;
 end = jndex;
 //OR: (you haven’t seen this in class/discussion,
 // but it came up in my office hours)
 //\*this = strView {index, jndex };
}