PIC 10A Disc 5A Midterm Review Worksheet

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

1. Assume you’ve already written code to make two strings called haystack and needle. Write code that finds **the longest run** of needle in haystack, represented as follows: the index where the longest repeated sequence begins, and the index *one after* the end of that copy of needle. (So if needle is “ab” and haystack “nabmababc”, you should say that the longest instance starts at index 4 and ends just before index 8.)
*Hint: loop over the indices for haystack. For each index, loop until you find a mismatch in needle.*
int const len=size(haystack), bunch=size(needle);
int start=0, longest=0;
for(int offset=0; offset < len; ++offset)
{
 int index;
 //Most of the time, the length left in the string won’t be an even multiple of bunch,
 //so we can optimize out the test. But sometimes the final ‘\0’s will coincide.
 for(index=0; (0 != len – offset % bunch) ||
 (index == len – offset); ++index)
 {
 if(haystack[offset + index]!=needle[index % bunch])
 {
 //Oops! Read too far. Back up and record.
 index -= index % bunch;
 break;
 }
 }
 if(index > longest)
 {
 //Copy over new run
 start = offset;
 longest = index;
 }
}
longest += start;
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:
int strComp(std::string const &needle, std::string const &haystack, int offset)
{
 int const len=size(haystack), bunch=size(needle);
 for(int index=0; (0 != len – offset % bunch) ||
 (index == len – offset); ++index)
 {
 int const shift=index % bunch;
 if(haystack[offset + index]!=needle[shift])
 {
 return index – shift;
 }
 }
 //Terminated by falling off the end of the string
 return len – offset;
}
//Main code becomes:
int start=0, longest=0;
for(int offset=0; offset < size(haystack); ++offset)
{
 int index = strComp(needle, haystack, offset);
 if(index > longest)
 {
 //Copy over new run
 start = offset;
 longest = index;
 }
}
longest+=start;
3. Define a new class:
class strView { public: int start, end; };
Make strComp operate on objects of type strView.

????????
4. Now make strComp a member function of strView.

??????????