HW16 – STL Algorithms & Function Objects [100 pts]

Answer the following questions by modifying the hw16.cpp source file and/or answering the question directly:

1. [read sgi.com – intro to stl, read wikipedia - stl, read learncpp.com – stl algorithms overview, read cplusplus.com - standard algorithms, read sgi.com – find algorithm] The *find* algorithm will find the first element in [first,last) that equals *value* where *first*, *last* are iterators. *find* returns an iterator pointing to *value* if a value is found or *last* if value has not been found. Recall *last* is one past the final element in a sequence. What kinds of operations must type InputIterator and type EqualityComparible support (i.e. what are the type requirements) such that *find* is able to work? Complete the implementation of the *find_improved* algorithm. Replace *for/if* logic in *find* with a *while* loop containing a logical && conditional. This will eliminate the temporary variable *p* improving the efficiency of the algorithm.
2. [read sgi.com – find_if algorithm] Rather than finding a sequence value directly, *find_if*, locates the first value which satisfies a specified condition. *find_*if uses a Predicate argument for the conditional. A predicate is a function which returns true or false depending upon a condition. Complete the implementation of the *find_if* algorithm. Hint: use logic from *find_improved;* this time sequence values are passed to the predicate for a conditional test (i.e. replace *!(*first==val)* with *!pred(*first)*). Complete the definitions for predicates: *even, less_than_31, less_than_v*.
3. [read sgi.com - function objects] Write a function object *Less_than* whose call operator compares argument *x* with class data member *v* of type int, *x < v*. *v* is initialized when class *Less_than* is instantiated.

4. [read sgi.com - sort algorithm, read sgi.com - function objects] Complete the implementation for the *Cmp_by_id* function object. *Cmp_by_id* compares two shapes by *id* via shape pointers (Shape *). Write additional function objects *Cmp_by_perimeter*, *Cmp_by_area* which compare pointers to shapes (Shape *) by *perimeter* and *area* respectively.
5. [read sgi.com - copy algorithm, read sgi.com - function objects] The *copy_if* algorithm requires type parameters Input_iterator<In>, Output_iterator<Out>, Predicate<Pred, Value_type<In>>. Explain what this means in terms of operations *copy_if* type parameters must support. What operations must these types have such that *copy_if* will work? Referring to *copy* complete the implementation of the *copy_if* algorithm. Include an *if* statement with a conditional based upon whether the predicate is satisfied to decide if a copy is made (i.e. *result++ = *first only if predicate is true). Complete the definition of the *Range_low_to_high* function object. The call operator() will only return true when element is >= low and <= high. Note: *Range_low_to_high* requires LessThan_comparable<Val> meaning the range must be determined by using operator <. Refer to *LessThan_comparable* equivalence semantics.

Include comments in your code to indicate which code segment answers which question. Appended written answers to the bottom of the hw16.cpp source file (as source comments via //).

Use the command script to capture your interaction compiling and running the program, including all operations, as shown below:

CS1C Spring 2021 TTH HW16 100 pts **Due: Tu 5/18/2021**

cs1c@cs1c-VirtualBox ~/cs1c/hw/16 $ script hw16.scr

HW16 – STL Algorithms & Function Objects [100 pts]

Script started, file is hw16.scr
cs1c@cs1c-VirtualBox ~/cs1c/hw/16 $ date
...
cs1c@cs1c-VirtualBox ~/cs1c/hw/16 $ ls -l
...
cs1c@cs1c-VirtualBox ~/cs1c/hw/16 $ make all
...
cs1c@cs1c-VirtualBox ~/cs1c/hw/16 $ ls -l
...
cs1c@cs1c-VirtualBox ~/cs1c/hw/16 $ ./hw16

... // print out output from steps 1 thru 5

cs1c@cs1c-VirtualBox ~/cs1c/hw/16 $ exit
Script done, file is hw16.scr
cs1c@cs1c-VirtualBox ~/cs1c/hw/16 $ make tar
...
Submit the tar package file hw16.tar by Tuesday May 18 2021.