ECE 218

Fall 2020

Exam 1 – Take Home

(open book / open notes)
This is an individual exam. You must complete it on your own. Any questions should be directed to the course instructor.

Name: _Jing Ma_____________________ **Student No.:** __C12108004_______________

Honor Code: On my honor, I have neither given nor received any aid on this exam

Jing Ma
(Please sign)

All code in the exam must be implemented using C++. You cannot use the C++ Standard Template Library.

Make sure that you include you design steps along with your code and sample results for each stage.

Submit your report as a PDF and include a signed version of the cover page.

For this system, you are implementing a utility application for the delivery drivers for a food delivery service. This service delivers ready to eat meals each packaged into individual boxes. The drivers are only allowed to deliver if the customer is present to accept the delivery and pay the outstanding balance. Otherwise it counts as a failed delivery.

The data file, `e1data.txt`, contains 1000 customers along with the number of boxes they ordered and their current balance. Each line contains:

```
   firstname lastname number_of_boxes balance
```

Included with this document are two support functions:

`randomInRange(start,end)` – a random number generator,
`getCPUTime()` – a timing function.

These are declared in `support.h` and implemented in `support.cpp`. These files are setup for proper compilation on either Unix or Windows.

The random number generator generates and returns a uniformly distributed random integer between start and end inclusive.

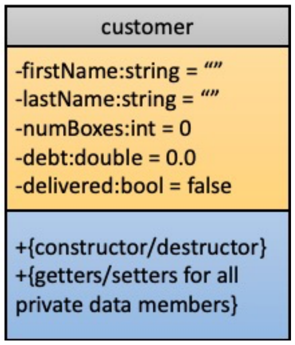
The timing function returns a double which represent the amount of CPU time the program has used up to when the function is called. To time how long some operation takes, simply use:

```
      double stime = getCPUTime();
      <operation(s) you want to time>
      double etime = getCPUTime();
      double elapsedTime = etime – stime;
```

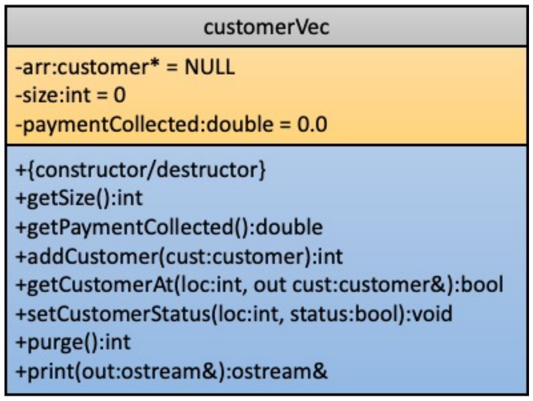
Now for the application:

Support Classes:

1. (10pts) Create and implement customer class according to the following model.



2. (40pts) Create and implement a dynamic array-based vector customerVec class according to the following model. Note DO NOT pre-allocate space.


customerVec
-arr:customer* = NULL
-size:int = 0
-paymentCollected:double = 0.0
+{constructor/destructor}
+getSize():int
+getPaymentCollected():double
+addCustomer(cust:customer):int
+getCustomerAt(loc:int, out cust:customer&):bool
+setCustomerStatus(loc:int, status:bool):void
+purge():int
+print(out:ostream&):ostream&

   a. addCustomer – adds a customer to the end of the vector (making space as needed)
   b. getCustomerAt – returns the customer at location loc in the array
   c. setCustomerStatus – set the customer delivered status
   d. getPaymentCollected() – sums, saves and returns the total payments received for all customer whose delivered status if True.
   e. purge() – purges all customers whose delivered status is True. Find an efficient way to do this.

Main Program:

3. (20pts) Create an application that:
   a. Creates an instance of customerVec.
   b. Reads in the data from e1data.txt and populates it into the customerVec.
   c. Time how long it takes to populate the vector and print this out.
4. (30pts) To simulate the delivery operation:

a. Go through each customer in the customerVec and randomly set their delivery status to True or False (make the True status 9 times more likely than the False i.e. 9 times out of 10 it chooses True).
b. Calculate the total payments collected and print this out.
c. Purge the vector of the customers whose delivery status is True. Time how long it takes to do this operation and print this out.