UMKC
School of Computing and Engineering

Assignment V,
Deadline: Sunday, November 5th, Midnight

Assignment learning objectives: This assignment is a tool to measure your knowledge in understanding and implementing the concept of *Polymorphism* using *virtual functions (late binding)* in C++.

The assignment problem: An appliances store needs your help to build a simple application that reads a shipping manifest of the newly arrived appliances at the store and prints a list of them on the computer screen. You do not know all types of appliances that this store receives. Currently, the store receives and sells only ovens, dishwashers, and refrigerators, but plans to extend in the future and add more appliances. Therefore, your application must be able to easily handle new types of appliances that will be added in the future with minimal code change (and utilize code reusability). *This is a scenario where you need to use the concept of polymorphism and pointers to handle different types of classes using their base class.*

Input File: You are given an input file containing the most recent shipping manifest from the factory. It contains the data for an unknown (to your program) number of appliances (you can assume there are no more than 100.) Each begins with a single upper-case letter; 'R' if the appliance is a refrigerator, 'D' for a dishwasher, 'O' for an oven. This is followed by the appliance's data, which varies depending on its type.

Your application must have a base class, *Appliance*, with two private data members, *modelNumber* and *serialNumber*, both are string types. Each of these data members has setters and getters. **Derive three classes from this base class as follows:**

- ***Refrigerator***. A refrigerator has a capacity in cubic feet (a float, default value 0), and may or may not be frost-free (a boolean, default value false). The boolean value is stored in input and output as 'T' for true and 'F' for false.
- ***Dishwasher.*** A dishwasher has a cycle time in minutes (integer) and has options for water saver mode or optional high-temperature dry (both booleans, input/output as 'T' or 'F')
- ***Oven.*** An oven has a size in cubic feet (a float) and may or may not be self-cleaning (stored as a bool, input/output is 'S' for self-cleaning, 'R' for regular) or have a convection option (another boolean, input/output as 'C' if convection, 'R' for regular).

<u>Virtual Functions:</u>

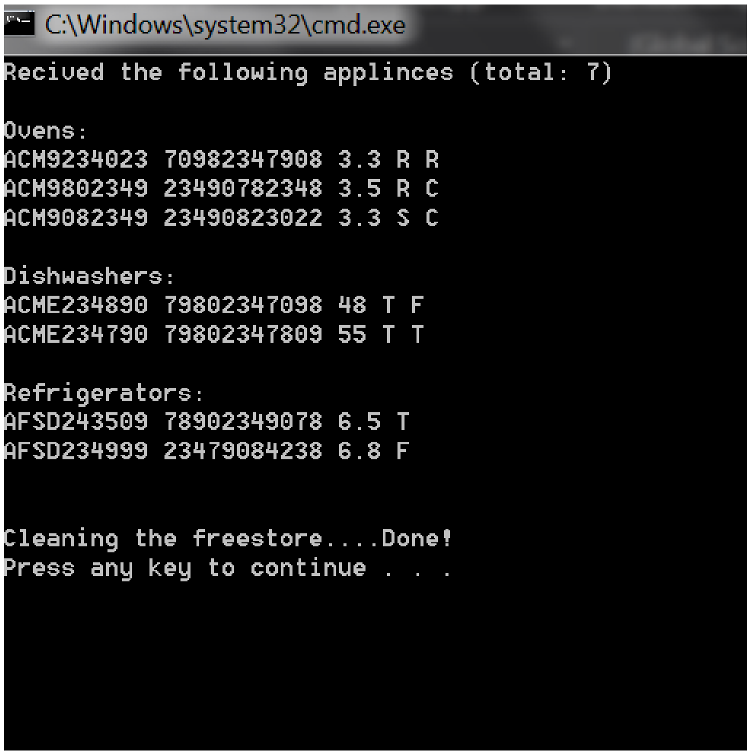
- Each appliance has a ***ReportType() const*** method that returns a string: 'Refrigerator', 'Dishwasher', or 'Oven'. This should be defined as a pure virtual function in the parent (Appliance) and separately written for the child classes.
- Each appliance has method **ReadData(istream&)** to read its data. This method is a void method. This must be defined in the parent class as a pure virtual function and implemented in each of the child classes.
- Each appliance has void method ***WriteData() const*** that writes the data of the appliance. This must be defined in the parent class as a virtual function and implemented in each of the child classes. The parent class definition prints the model and serial numbers of the appliance; and must be called in each of the child classes' definitions to print the serial and model numbers before printing the other information.

<u>Main application:</u>

Your main program should have an array of 100 pointers to Appliance, called *InStock []*. As you read the data file, begin by reading the object's type. Based on the type, create a dynamic object of that type and store it in the array, InStock[], (Hint: you can use *switch-case*.) Then call the ReadData() method using the new object to get the data from the file for that object. After reading the entire file, call WriteStuff() function (more details below), to print the list of appliances on the screen grouped by their type. See figure 1.

Write a `void WriteStuff(Appliance* items[], int Count, const string& TargetType)` function in the main file to print on the screen the appliances stored in the array, each appliance printed based on its type, by calling the WriteData() function. This should be done by iterating over the array of pointers three times and using the ReportType() method to select only one type of appliance at a time.

After printing the list, delete all of the dynamically created variables using a loop at the end of the main function.



```
Recived the following applinces (total: 7)

Ovens:
ACM9234023 70982347908 3.3 R R
ACM9802349 23490782348 3.5 R C
ACM9082349 23490823022 3.3 S C

Dishwashers:
ACME234890 79802347098 48 T F
ACME234790 79802347809 55 T T

Refrigerators:
AFSD243509 78902349078 6.5 T
AFSD234999 23479084238 6.8 F


Cleaning the freestore....Done!
Press any key to continue . . .
```

Figure 1. Sample output for the given input file.

Zip the entire project and submit it no later than Sunday, November 5th at midnight.