CIS 22B Assignment 4

Topics:

Arrays and classes

Use the same format for problem and function headings as assignment 1

Problem 4.1

In this problem, we will read cars from a file, rather than typing them in from the keyboard. Do the steps in order to build the solution to this problem.

1. Copy and clean up code from problem 3.3
   * Copy problem 3.3
   * Change the name to Assignment 4, Problem 4.1
   * Remove everything from the main function.
2. Modify the input function.
   * Remove the & from the parameters in the function header, so they are all values rather than references.
   * Move the parameters from the function header and put them within the input function, so they are now all local variables.
   * Remove the parameters from the prototype for the input function, so it matches the function header.
   * At the bottom of the input function, declare a Car object named temp using the constructor that takes the five parameters.
   * Use the Car output function to print the Car.
   * Call the input function from the main function with no arguments.
3. Create a file and use it for input. This is good because we will be using the input many times.
   * Use the provided **cardata.txt** file which contains the following three lines of data (omit the heading line)

```
Type  ARR  number  kind         loaded  destination        OMIT THIS LINE

Car   CN   819481  maintenance  false   NONE
Car   SLSF 46871   tank         true    Memphis
Car   AOK  156     tender       true    McAlester
```

   * In the input function, declare an object of type ifstream named inputFile, which we will use to read from the file.
   * At the beginning of the code for the input function, open the file. If the open fails, send a message to stderr and exit the program.

* In all the reads within the input function, remove the user prompt and read from the inputFile object, rather than reading from the stdin object.
* In the input function declare a string named type. Read the Type field before reading the ARR field. We do not need the Type field yet, but we need to read it to get it out of the way.
* **Hint: You need to use getline when reading the destination.** using >> skips leading white space before reading the data. getline does not skip this leading whitespace. So, before using getline use the following code:

```
while(inputFile.peek() == ' ')
  inputFile.get();
```

peek looks at the next character you are about to read. If it is a space, get is used to read the space character, to get it out of the way.
* Use a loop to read each line from the file. To do this use a while loop including all the reading in the input function, as well as the building and output of the Car.
Hint: you can do this with the following while statement:

```
while(inputFile.peek() != EOF)
```

The peek function will return EOF is there is no next character.
* At the bottom of the input function, close the file.

Order of functions in the code:

1. main
2. Car member functions
   1. constructors in the order
      1. default constructor
      2. copy constructor
      3. other constructors
   2. output
   3. setUp
3. operator== with Car parameters
4. input

Put an eye catcher before the beginning of each function, class, and the global area:
// class name function name comment(if any)
****

Problem 4.2

Copy the solution for problem 4.1

Copy the following operator= overload member function that returns the
left hand operator by reference:

```
// Car  operator=  ************************************************
Car & Car::operator=(const Car & carB)
{
 reportingMark = carB.reportingMark;
 carNumber     = carB.carNumber;
 kind          = carB.kind;
 loaded        = carB.loaded;
 destination   = carB.destination;

 return * this;
}
```

Several cars coupled together are referred to as a string of cars.

Create another class called StringOfCars, containing:

* a pointer to an array of Car objects in the heap.
* a static const int ARRAY_MAX_SIZE set to 10.
* an int size containing the current number of Cars in the array.
* a default constructor which gets space for the array in the heap,
  and sets the the size to zero.
* a copy constructor which gets new space in the heap for the array
  and copies all the Car objects.
* a destructor which returns the space to the heap.
* a push function which adds a car to the string of cars.
* a pop function which removes a car from the string of cars, Last In
  First Out (LIFO).
* an output function which prints a heading for each car:
  car number n where n is the position in the array starting from 1
  for the first car and then uses the Car output function to print the
  data for each car Or, if the array is empty prints: NO cars

Order of functions in the code:

1. main
2. Car member functions
   1. Car constructors in the order
      1. default constructor
      2. copy constructor
      3. other constructors
   2. output
   3. setUp
   4. operator=

3. StringOfCars member functions
   1. Car constructors in the order
      1. default constructor
      2. copy constructor
   2. destructor
   3. output
   4. push
   5. pop
4. operator== friend of Car
5. input

Put an eye catcher before the beginning of each function, class, and the global area:
// class name function name comment(if any)
****

Modify the main function to do the following tests:

1. Test the Car operator= function.
   Before the call to the input function:
   Print: TEST 1
   Creat a Car object named car1 with initial values:
    reportingMark: SP
    carNumber: 34567
    kind: box
    loaded: true
    destination: Salt Lake City
   Create a Car object named car2 which is a copy of car1
   Print car2
2. Test the StringOfCar push function.
   Change the input function to have a parameter that is a reference to a StringOfCars.
   Add to main:
   Create a default StringOfCars object named string1.
   Print: TEST 2
   Pass string1 to the input function.
   Use the same **cardata.txt** file as in problem 4.1
   In the input function, just after creating the car, push it on to string1.
   Remove the print of the car in the input function.
   Print: STRING 1
   In the main function, after using the input function, print string1.
3. Test the StringOfCars pop function.
   Add to main:
   Print: TEST 3
   Create a car named car3.
   Pop one car from string1 into car3.
   Print: CAR 3
   Print car3.
   Print: STRING 1

Then print the contents of string1 again