Project 1 Pseudocode: Anja Sheppard AMS180001

**#define statements for the dimensions of drinks**

**enum customerType**

* Two types: REGULAR and PREFERRED

**struct orderData**

**Variables:** string customerID, enum custType, char size, string drinkType, float squareInchPrice, int quantity, bool valid

**parseCustomerInfo**

**Parameters:** array of Customers

**Returns:** nothing

* Open file “customer.dat”
* While loop until getline() reaches EOF
  + Parse each line (based on spaces) for customer ID, First Name, Last Name, and Amount Spent
  + instantiate the customer object with these parameters
  + Add new object to array with add function
* Close file

**ParsePreferredCustomerInfo**

**Parameters:** array of Customers

**Return:** nothing

* Open file “preferred.dat”
* If the file is empty (or just has whitespaces)
  + Close file
  + Return and exit function
* Else
  + while loop until getline() reaches EOF
    - Parse each line (based on spaces) for Customer ID, First Name, Last Name, Amount Spent, and Discount/Bonus Bucks
    - If Discount/Bonus Bucks value has a ‘%’ char at the end
      * Instantiate PreferredCustomer (gold) object with parameters extracted above
    - Else
      * Instantiate PreferredCustomer (platinum) object with parameters extracted above
    - Add new object to array using add function
* Close file

**parseOrders**

**Parameters:** two arrays of customers (one regular one preferred)

**Return:** nothing

* Open “orders.dat”
* While loop until getline() reaches EOF
  + Run validData function and store struct in variable
  + If orderData.valid is true:
    - Calculate amount spent (disregarding any discounts as those will be applied below) via amountSpent function
    - If orderData.customerType is REGULAR:
      * If that total + previous total is greater than or equal to $50 and less than $200:
        + update Customer to PreferredCustomer (gold) by creating new object and passing the existing variable values through the constructor
        + Resize preferredCustomers array via add() in order to append this new PreferredCustomer
        + Resize the regularCustomers array via remove() in order to remove the upgraded Customer
        + If amount spent (previous + current) is between $50 and $100

update discount percentage to 5 and apply a 5% discount to the total, then add the total to the previous total

* + - * + Else if the amount spent (previous + current) is between $100 and $150

Update the discount percentage to 10% and apply this discount to the total, then add the total to the previous total

* + - * + Else if the amount spent (previous + current) is between $150 and $200

Update the discount percentage to 15% and apply the discount to the total, then add the total to the previous total

* + - * Else if the total + previous is less than $50:
        + Update the total spent
      * Else if the total + previous is greater than $200:
        + Update the total spent
        + update PreferredCustomer (gold) to PreferredCustomer (platinum) by creating new object and passing the existing variable values through the constructor
        + The number of bonus bucks is equal to (totalSpent – 200) / 10
    - Else if the customer is a preferredCustomer (gold):
      * If the previous total + this order’s total is now between $100 and $150:
        + Update discountPercentage to 10%l
      * Else if the previous total + this order’s total is now between $150 and $200
        + Update discountPercentage to 15%
      * Update total spent with the discount percentage
      * If the total spent is more than $200, update the preferredCustomer (gold) object with a platinum object
        + If the total spent is above $210, then check how many bonus bucks the customer should receive and update that value
    - Else if the customer is a platinum preferred customer:
      * Update total + previous (include deduction from existing bonus bucks)
      * Update number of bonus bucks
  + Else (the data is not valid)
    - Do nothing with this line

**remove**

**Parameters:** array of customer objects, int index to be removed

**Return:** nothing

* Create new array with a size of the previous array’s size - 1
* For loop through the old array
  + copy each object over to the new array, skipping the object to be deleted (at the indicated index)

**add**

**Parameters:** array of customer objects, customer object to append

**Return:** nothing

* Create new array with a size of the previous array’s size + 1
* For loop through the old array
  + copy the objects over to the new array
* At the end of the new array, add in the new object

**validData**

**Parameters:** string line (from the order file)

**Return:** orderData struct

* Create orderData struct, set all values to NULL
* Check that there are characters and a space in the line string
  + take first set of characters (before the first space), delete them from the line string, and proceed to check if that ID exists:
  + For loop through regular customer array
    - If current customerID matches the one found in the file
      * Set orderData.customerID to the substring
      * Set orderData.customerType to REGULAR
  + If orderData.customerID is NULL and preferred Customer array exists:
    - For loop through preferred Customer array
      * If current customerID matches the one found in the file
        + Set orderData.customerID to the substring
        + Set orderData.customerType to PREFERRED
  + If orderStruct.customerID is still NULL
    - Set orderData struct valid bool to false
    - Return orderData struct
* Else
  + Set orderData struct valid bool to false
  + Return orderData struct
* Check that there is just one character and a space
  + Take the first set of characters (before the first space) and delete them from the string
  + If size doesn’t equal ‘S’, ‘M’, or ‘L’ (or those lowercase)
    - Set orderData struct valid bool to false
    - Return orderData struct
  + Else
    - Set orderData.size equal to the substring converted to a char
* Else
  + Set orderData.valid to false
  + Return orderData struct
* Check that there are characters and a space
  + Take the first set of characters (before the first space) and delete them from the string
  + If drinkType doesn’t equal “soda”, “tea”, or “punch” (or those uppercase)
    - Set orderData struct valid bool to false
    - Return orderData struct
  + Else
    - Set orderStruct.drinkType equal to the substring
* Else
  + Set orderData.valid to false
  + Return orderStruct
* Check that there are characters and a space
  + Take the first set of characters (before the first space) and delete them from the string
  + If a float is returned and no exception is thrown while converting the substring to a float using stof() function
    - Set orderData.squareInchPrice to the float
  + Else
    - Set orderData.valid to false
    - Return orderData struct
* Else
  + Set orderData.valid to false
  + Return orderStruct
* Check that there are characters and a newline (or space then newline)
  + Take the remaining characters and put them into a substring
  + If an int is returned and no exception is thrown while converting the substring to an int using stoi()
    - Set orderData.quanitity to the int
  + Else
    - Set orderData.valid to false
    - Return orderData
* Else
  + Set orderData.valid to false
  + Return orderStruct

**amountSpent**

**Parameters:** string drinkType, char size, float squareInchPrice

**Returns:** float (the money spent in that order)

* Calculate base price (based on size and drinkType)
* If a graphic is included
  + Calculate surface area of beverage and multiply by the squareInchPrice
  + Add that to the base price of the drink
* Return drink price

**writeToFile**

**Parameters:** regular customer array, preferred customer array

**Return:** nothing

* Open file “customer.dat”
* For loop for the regular customer array
  + Write in each customer with the appropriate format
* Close file “customer.dat”
* open “preferred.dat”
* For loop through the preferred array
  + Write in each preferred customer with the appropriate format
* Close the file

**main.cpp**

* Create regularCustomers array and preferredCustomers array
* read “customer.dat” into regularCustomers array via parseCustomerInfo function
* If “preferred.dat” exists
  + read data from file into array via parsePreferredCustomerInfo function
* Run parseOrders function
* Run writeToFile function