Western Washington University
Computer Science Department

CSCI 141 Computer Programming I
Fall 2011

Assignment 3

Submitting Your Work

This assignment is worth 15% of the grade for the course. In this assignment you will write an Ada program to maintain the inventory data for an on-line retail system. Submit your program file (the .adb file) in a zipped tar file Wnnnnnnnn.tar.gz (where Wnnnnnnnn is your WWU W-number) and submit the file via the **Assignment 3 Submission** item on the course web site. You must submit your assignment by 4:00pm on Friday, December 2, 2011.

Inventory items

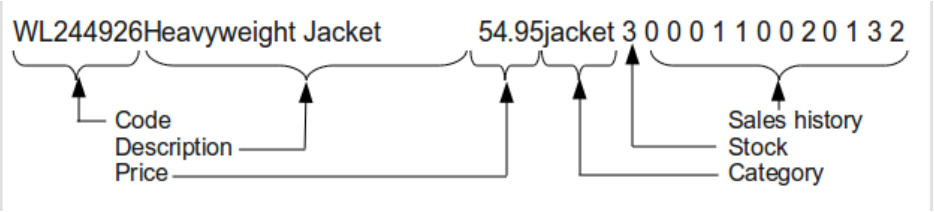
The on-line retail system has an inventory of products which are offered for sale. For each product, the inventory system keeps track of:

- Product code: a string of 8 characters
- Product description: a string of 30 characters
- Price: in dollars, accurate to the nearest cent, for example 23.45
- Product category: one of jacket, shoes, shirt, sweater, pants, accessory
- Stock on hand: an integer number greater than or equal to 0
- Sales history: number of items sold in each of the preceding 12 months. The first number in this sequence represents the total sales quantity for the most recent month.

Inventory File

The current state of the product inventory is saved in the file `inventory.txt`. Each line in that file represents one product. The values of the data components for that product are listed on the line in the same order as the components are listed in the above description of the inventory items.

For example, the first line in the file is:



Transactions

Transactions are events which cause a change to the inventory data. There are five types of transaction in this system:

- Sale: a quantity of one product has been sold and is removed from the stock for that product.
- Delivery: a quantity of one product has been delivered and is added to the stock for that product.
- New product: a new product is added to the inventory.
- Delete product: an existing product is deleted completely from the inventory.
- Price change: the price of one product is changed.

Transaction File

The transactions for the last month are provided in the file `transactions.txt`. Each line in that file gives the data for one transaction. There may be more than one transaction for any product. The data given for a transaction depends on the type of transaction.

- Sale: product code and quantity sold. For example:

  ```
  SALE WL237870 2
  ```

  This represents a sale of 2 units of the product with code WL237870.

- Delivery: product code and quantity delivered. For example:

  ```
  DELIVERY WL242373 10
  ```

  This represents a delivery of 10 units of the product with code WL242373.

- New product: product code, description, price and category of a new product. For example:

  ```
  NEWPRODUCT WL220312New Wave Jacket                 55.50 jacket
  ```

  This represents a new product with product code WL220312, description "New Wave Jacket" (padded out with spaces to 30 characters), price $55.50, and category jacket.

- Delete a product: product code. For example:

  ```
  DELETE WL036147
  ```

  This means that the product with code WL036147 is to be deleted from the inventory.

- Price change: product code and new price. For example:

  ```
  PRICE WL162343 45.50
  ```

  This indicates that the price for the product with code WL162343 is to be changed to $45.50.

Note that there is a space between the transaction type (SALE, DELIVERY, NEWPRODUCT, DELETE or PRICE) and the product code. Your program needs to read that space before reading the product code.

The transactions are listed in the file in the order in which they occurred during the last month. Therefore, if there is a price change for a product and two sales transactions for that same product, one before the price change and one after the price change, the first sale will be valued at the old price and the second sale will be valued at the new price.

What your program must do

1. Read the current inventory data from the file inventory.txt into an array of records. Each record in the array is to hold the data for one product. You must allow for up to 200 products.

2. Read the transactions from the file transaction.txt and apply each transaction to the inventory data held in the array.

3. For a new product, the sales history is to be set to zero for each month.

4. The total sales quantity for the month is to be added as the first component of the sales history, pushing all the existing history items along one place and discarding the last number from the sequence.

5. Check for errors in the transactions. If any error is found, display a message on standard output with the details of the transaction and the nature of the error. A transaction containing any error is not to be used to update the inventory data.

   The type of error to look for depends on the type of transaction.

   - For sale, delivery, delete and price change transactions, report an error if the transaction product code does not exist in the inventory.

   - For a new product, report an error if a product with that code already exists in the inventory.

   - For a sale, report an error if the sale quantity is larger than the stock of that product in the inventory.

6. After all the transactions have been processed, write the updated inventory data to a file called `newinventory.txt`.

7. After all the transactions have been processed, display a report on standard output showing the total sales value (total dollar amount) for each category for the month represented by the transaction file. The categories are to be listed in descending order of total sales value. For example:

```
Category        Total sales
shirt           $327.50
shoes           $127.32
jacket          $109.90
sweater         $ 87.53
accessory       $ 53.90
```

What you must submit

You must submit the Ada source file (the .adb file) for the inventory control program. This file must be included in a zipped tar file.

Coding Standards

1. Use meaningful names that give the reader a clue as to the purpose of the thing being named.

2. Use comments at the start of the program to identify the purpose of the program, the author and the date written.

3. Use comments at the start of each procedure to describe the purpose of the procedure and the purpose of each parameter to the procedure.

4. Use comments at the start of each section of the program to explain what that part of the program does.

5. Use consistent indentation:

   - The declarations within a procedure must be indented from the Procedure and begin reserved words. The body of the procedure must be indented from the begin and end reserved words. Example of procedure indentation:

```
procedure DoStuff is
     Count : integer;
     Valid : boolean;
begin
     Count := 0;
     Valid := false;
end DoStuff;
```

   - The statements within the then-part, each elsif-part and the else-part of an if statement must be indented from the reserved words if, elsif, else and end.

```
if Count > 4 and not Valid then
     Result := 0;
     Valid := true:
elsif Count > 0 then
     result := 4;
else
     Valid := false;
end if;
```

- The statements within a loop must be indented from the loop and end loop.

```
loop
     Count := Count + 1;
     Get (Number);
     exit when Number < Count;
end loop;
```

- The exception handlers and statements within each exception handler must be indented.

```
begin
     ...  -- normal processing statements
exception
     when Exception1 =>
          Put_Line (“An error has occurred”);
          Total := 0;
     when others =>
          Put_Line (“Something weird happened”);
end;
```