# Western Washington University
## Computer Science Department

### CSCI 145 Computer Programming and Linear Data Structures
### Winter 2012

### 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 for inventory management of a book store. Save your .adb file in the zipped tar file WnnnnnnnnAssg3.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 10:00pm on Friday, March 9.

## Book Store Inventory System

A book store stocks a number of book titles. The provided file Books.txt gives the details of the inventory, one book per line, giving the ISBN, price and current stock level.

Transactions for the inventory system are provided in the file Transactions.txt. Two types of transactions have to be handled by the inventory system:

- STOCK transactions, each of which specifies the ISBN of a book, the unit price of the book in dollars, and the number of copies of that book to be added to the stock level.
- ORDER transactions, each of which specifies the ISBN of the book being ordered, the number of copies of the book required by the customer, and a 6-digit customer number.

If an order cannot be completely filled, because of insufficient stock, the unfilled part of the order (the entire order if there is zero stock) must be placed on a queue of back orders for that book.

When new stock is received for a book, the inventory system must work through the back order queue for that book, filling as many back orders as it can. If a back order can only be partly filled before the stock level of the book drops to zero, the remainder of that back order must remain at the head of the back order queue for that book.

You must use the provided generic queue package for your back order queues.

## Program Requirements

1. Your program must get the names of the book file and the transaction file from the command line. If fewer than 2 command line arguments are provided, the program must display a usage message and terminate.

2. If either of the input files cannot be opened, the program must display an appropriate message and terminate.

3. Your program must display a single line report for each inventory event, for example:

```
Stock for 0929701836 increased from 0 to 15

Order filled for customer 558432 for 12 copies of book 0929701836

Order filled for customer 659023 for 1 copy of book 0929701836

Back order of 1 copy of book 0929701836 for customer 659023

Back order of 3 copies of book 1565129520 for customer 605432
```

## What you must submit

You must submit the Ada source file for the book store inventory system (the .adb file). This files must be included in a zipped tar file.

## Saving your files in a zipped tar file

Use the command:

```
tar -cf WnnnnnnnnAssg3.tar bookstore.adb
```

(where Wnnnnnnnn is your W-number).

The -cf specifies two options for the tar command: 'c' means create and 'f' means that the name of the resulting tar file comes next in the command.

If you now use the command `ls` you should now see the file WnnnnnnnnAssg3.tar in your directory.

If you use the command

```
tar -tf WnnnnnnnnAssg3.tar
```

(where Wnnnnnnnn is your W-number), it will list the files within the tar file.

Now compress the tar file using the gzip program:

```
gzip WnnnnnnnnAssg3.tar
```

By using the `ls` command again, you should see the file WnnnnnnnnAssg3.tar.gz in your directory. This is the file that you need to submit through the **Assignment 3 Submission** link in the moodle web site.

## Coding Standards

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

2. Use named constants instead of repeated use of the same numeric constant.

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

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

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

6. 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;
```