UNIVERSITY OF WOLLONGONG IN DUBAI

Assignment Cover Sheet

Subject Code: CSCI251
Subject Name: Advanced Programming
Submission Type: Report
Assignment Title: Assessment 03 Report
Student Name: Akshita Bhatia
Student Number: 7080116
Student Phone/Mobile No.: 056 4687028
Student E-mail: asb272@uowmail.edu.au
Lecturer Name: Prof. Lim H.C
Due Date: 3rd December, Saturday @10pm
Date Submitted:

PLAGIARISM:	**DECLARATION:**
The penalty for deliberate plagiarism is FAILURE in the subject. Plagiarism is cheating by using the written ideas or submitted work of someone else. UOWD has a strong policy against plagiarism. The University of Wollongong in Dubai also endorses a policy of non-discriminatory language practice and presentation. **PLEASE NOTE:** STUDENTS MUST RETAIN A COPY OF ANY WORK SUBMITTED	I/We certify that this is entirely my/our own work, except where I/we have given fully documented references to the work of others, and that the material contained in this document has not previously been submitted for assessment in any formal course of study. I/we understand the definition and consequences of plagiarism. **Signature of Student:**

Optional Marks:

Comments:

Lecturer Assignment Receipt (To be filled in by student and retained by Lecturer upon return of assignment)

Subject:	**Assignment Title:**
Student Name:	**Student Number:**
Due Date:	**Date Submitted:**
Signature of Student:	

Student Assignment Receipt (To be filled in and retained by Student upon submission of assignment)

Subject:	**Assignment Title:**
Student Name:	**Student Number:**
Due Date:	**Date Submitted:**
Signature of Lecturer	

Index

Design Considerations & Flow Processes

Abstract:

The idea was to create a C++ code that can:

a) Have a dynamic menu (i.e. can move from one menu to another)
b) Give options to read and display data from the 3 test data files
c) Display the data according to the specific format, values, and denominations

Flow/Algorithm Process:

The First Step:

Was to create a menu (as this project needs to be menu driven) to give the user the option to read the data either from Test Data1, Test Data2 or Test Data3, then create a code that can read from the file.
In the beginning of this project, I have mainly used '***fstream***' to read the data line by line, which worked! However, when it came to storing the values from that data, it became complicated.
For storing, I have used '***vectors***', initially I had one ***string vector*** to store the data from the file…and I test it by printing it using '***cout***', it gave me an error during run time, '**vector argument is too long**' or '**vector beyond subscript**' (look at the picture below).

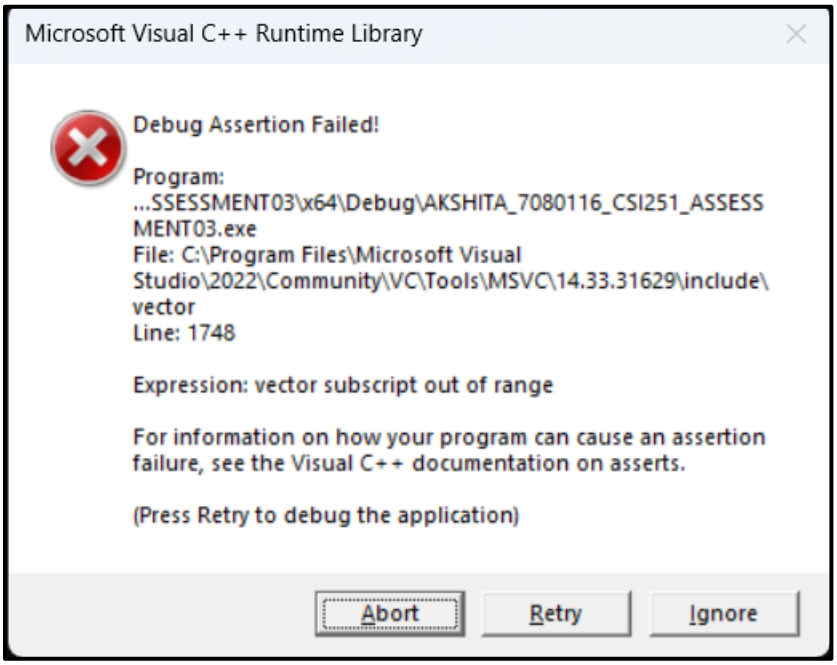

Figure 1.1

The reason this happened was, when storing the data each value is assigned to a particular index of the vector, e.g. "**Sardines**", "**Apples**" and so on were **vector[0]**, and etc. This caused a problem, especially for unit cost and quantity and the numeric value that payment was made in, because they need to be stored as double for calculation purposes and not all test data files have the same type of data, e.g. test data 1 has an exact payment, that means no payment value given, whereas test data 2 & 3 have a payment value.

To solve this, I decided to read the file in 2 ways,

1) read all lines except the last line – for storing item details

```
//READING ALL LINES EXCEPT THE LAST

ifstream ifs(filename);

if (!ifs.is_open())
     (cout << "Cannot open file\n"), 1;

ifs >> noskipws;

string data((istream_iterator<char>(ifs)), istream_iterator<char>());

data.resize(data.find_last_of('\n'));

istringstream iss(data);

conversion c1;

c1.print();

for (string line; getline(iss, line); )
{
    Item i; // Storing item details
    i.item(line);

    c1.total_purchase(i.total_item_cost()); // total purchase in dhs

    c1.get_vector(i.return_vector());

    c1.get_total_cost(i.total_item_cost());

    c1.printbill();
}
```

Figure 1.2 [1]

For this, I have used, '***iterators***', and '***resize***', to find the last of newline, once it finds it, the **iterator** will resize and read from it accordingly.

2) read only the last line – for storing the payment type details

```
//READING THE LAST LINE

ifstream in(filename);

string l;

if (in)
{
    while (in >> std::ws && std::getline(in, l)); // skips the empty lines

    payment p; // Storing item details
    p.pay(l);

    c1.convert_tp(p.currencyname());
    c1.paid_val(p.paid_values());
    c1.con_return_amount(p.currencyname());
}
else
{
    cout << "Unable to open file.\n";
}
```

Figure 1.3 [2]

The Second Step:

In *Figure 1.2* & *Figure 1.3*, you can see that some classes are called…
In this project, I have used 3 classes.

Class 1:- Item Class

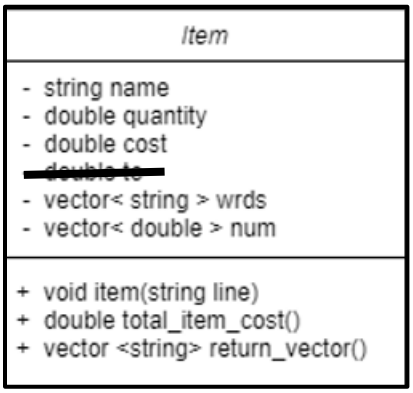

Figure 2.1

Item class is responsible for storing item details such as, item name, unit cost and quantity. There 2 vectors that are being used, one is to store the whole details as a string, and the other to store the numeric values as a double, in order to extract the numeric values and store it as double, I used [3] '***stringstream***', which allows you to read from the string as if it were a stream (like **cin**) and stored it in a '***vector <double>***', because there only 2 numeric values (cost & quantity), the **double variables** are stored like this, '***cost = num[0];***' and '***quantity = num[1];***', similar process for strings.

Class 2:- Payment Class

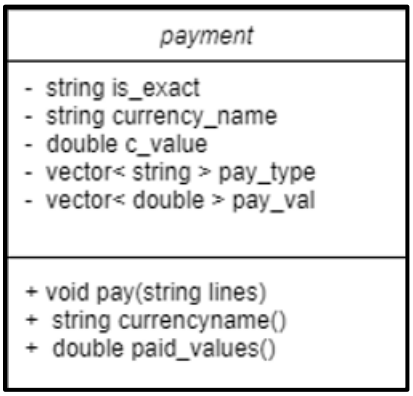

Figure 2.2

Payment class is responsible for storing the details of the last line that is being read from the file, e.g. '***Exact payment made in Dhs***', the process is similar to the one done in item class in regards of storing. However, unlike the item class, payment class is responsible for checking whether the payment is made in Dirhams, USD Dollars, or Euros, by using, '***Iterators***' and '***.find()***'.
After checking all this, the string variable '***currency_name***' will be assigned accordingly.

Class 3:- Conversion Class

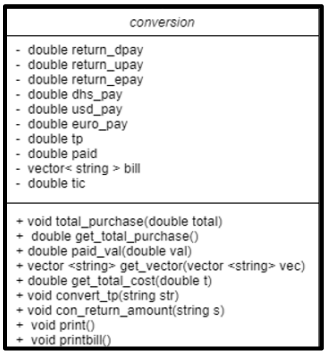

Figure 2.3

Conversion class is responsible for calculating **total purchase**, **return amount** and **displaying the output** with the correct values and symbols. It is also responsible for printing and populating the output in its specific format. In order to achieve that format, I have used '***setw()***', that is set width.
This also where the advance task will take place, which be discussed in page 6, *click here to check it out*.

Using OOP:- Object Oriented Programming

One of the requirements for this project was to use **OOP**, that is ***Object Oriented Programming***.
So I have decided to use [4] ***Hybrid Inheritance***, also known as ***multipath inheritance***, it is a combination of different types of inheritance such as, **single inheritance**, **multilevel inheritance**, **hierarchical inheritance**. This type of inheritance allows the child class to inherit from more than one parent class.
In this case (look at the picture below), class conversion inherits from class Item and class payment.

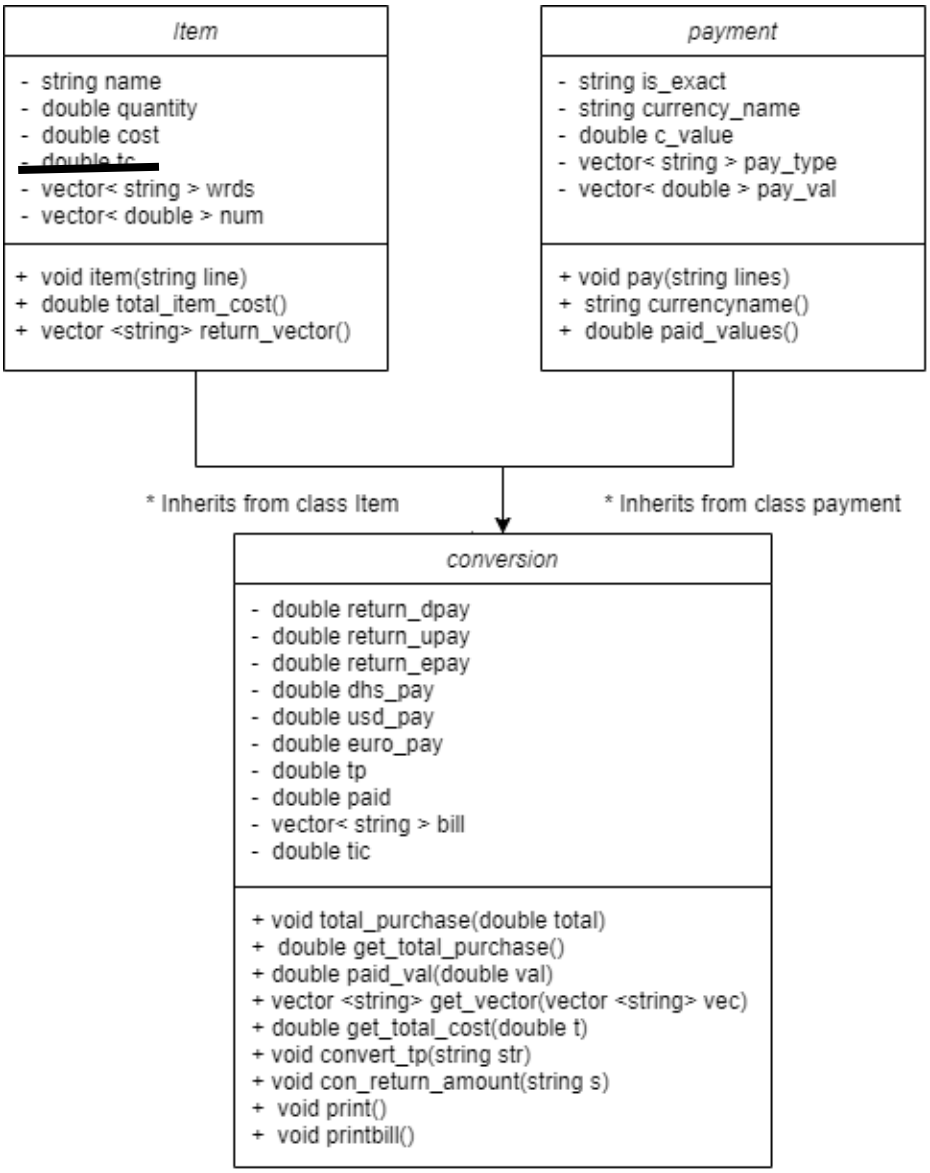

Figure 2.4

Advance Task

Assumption:

The advance task in this project is to allow or give an option to pay in Dirhams, Dollars, or Euros. Hence, I am **assuming** that the task is to ensure that if the payment is made in either **Dollars** or **Euros**, there should be a **conversion of the main currency** that is from **Dirhams** to either **Dollars** or **Euros**, and the output will display the **respective values** and **denominations**.

Task:

The advance task takes place in the **conversion class**, which is the **child class** of **Item class** and **payment class**.

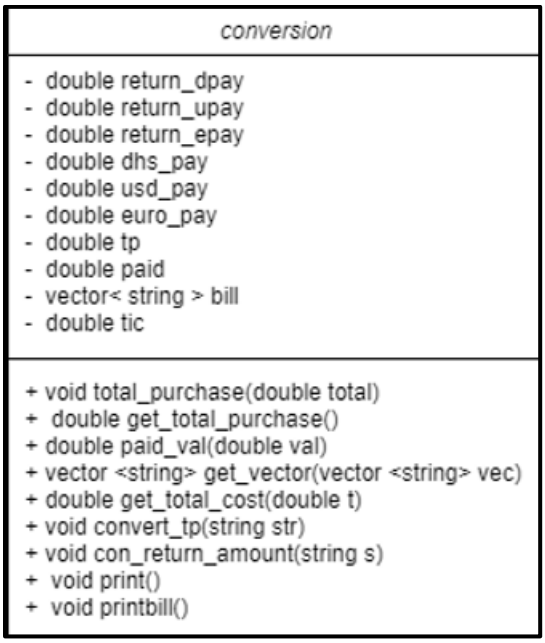

Figure 3.0

What this class does is, it takes variables such as **total item cost**, **item details** from item class and use it to calculate **total purchase** and **convert** it **into** its **respective currency** i.e. **Dhs, USD** or **Euros** and it will **populate the output** in the **function** called '***void printbill()***'. Similarly, it takes variables such as **currency_name**, **value that it's paid in** from payment class and use it to **check** if the payment is made in **Dhs**, **USD** or **Euros**, then it will **convert** in the **functions**; '**void convert_tp(string str)', ** converting **total purchase** and '**void con_return_amount(string s)', ** converting the **return amount** from **USD** or **Euros** respectively into **Dhs**. Then this will be **populated** in the **function** '***void printbill()***'.

The conversion in '**void convert_tp(string str)**' is:

USD - (total purchase in **dirhams**) * **0.27**
Euros - (total purchase in **dirhams**) * **0.26**

The conversion in '**void con_return_amount(string s)**' is:

USD - ((the **original value** that the payment was made in) * **3.67**) – ((total purchase in **USD**) * **3.67**)
Euros - ((the **original value** that the payment was made in) * **3.83**) – ((total purchase in **Euros**) * **3.83**)

Lessons Learnt

From this project I have **learned**:

- How to analyze and decompose the tasks that are given into smaller tasks that can be solved step by step
- How to create UML diagrams that help represent and visualize my classes better
- How to use header files and cpp files that helps to keep my classes organized
- How to use object-oriented programming, inheritance to make the code more efficient and neater
- How to incorporate STL in my code
- How to create a dynamic menu with the use of loops
- How to write a report that can sufficiently explain my code

References

[1] How to ignore the last line of a file - C++ Forum (cplusplus.com)

[2] Read last line text file - C++ Forum (cplusplus.com)

[3] stringstream in C++ and its Applications - GeeksforGeeks

[4] Hybrid inheritance in C++ - javatpoint