CE320-6-AU/AT

UNIVERSITY OF ESSEX

Undergraduate Examinations 2022

LARGE SCALE SOFTWARE SYSTEMS AND EXTREME PROGRAMMING

Time allowed: **TWO** hours (exam time) + **ONE** hour to allow for submission time (total **THREE** hours)
(Please see your exam timetable or check on FASER for the deadline to upload your answers)

The times shown on your timetable are in British Summer Time (BST) (GMT+1). Please check online for a conversion to your local time if you will be undertaking your assessment outside the United Kingdom

Candidates are permitted to use:
Calculator – Casio FX-83GT Plus/X or Casio FX-85 GT Plus/X ONLY

Candidates must answer **ALL** questions

The paper consists of **FOUR** questions

The questions are **NOT** of equal weight

The percentages shown in brackets provide an indication of the proportion of the total marks for the **PAPER** which will be allocated.

If you have a query with the content of this exam paper please use the revision FAQ Forum on the module's Moodle page. Your academic will be available to answer any queries in real-time.

If you have a technical problem with FASER, or any other query, please go to Exams Website to find contact details of the teams that can help you.

Please note that the time allocated for this assessment includes time for you to download this question and answer paper and to upload your answers to FASER.

You should complete the Answer Sheet supplied using a word-processed file format or PDF. If you need to handwrite/draw any or all of your answers, you should take a photo or scan these to be included **WITHIN** one answer sheet. (i.e. ALL YOUR ANSWERS SHOULD BE UPLOADED TO FASER IN ONE DOCUMENT – THE EXTRA HOUR ON THE EXAM LENGTH IS TO ALLOW YOU TIME TO DO THIS AND DEAL WITH ANY TECHNICAL ISSUES).

Please allow at least 30 minutes at the end of your exam time to upload your work. Once you have completed the assessment do not leave it to the last minute to upload.

Please save your work throughout the examination to avoid losing your work. Please do not communicate with any other candidate in any way during this assessment. Your response must be your own work. Procedures are in place to detect plagiarism and collusion.

Question 1

Two practices are traditionally used to try to get a late software project back on track: hiring more programmers and asking existing programmers to work for longer hours.

a) List the drawbacks of hiring more programmers. [9%]

b) List the drawbacks of asking existing programmers to work for longer hours. [7%]

c) Indicate how the problem of schedule slips is addressed in extreme programming. [9%]

Question 2

Unit testing is one of the key practices in XP.

a) Explain what unit testing is. [5%]

b) Explain the role of unit testing in test-driven development (TDD). [10%]

c) Explain the role of unit testing in refactoring. [5%]

Question 3

You are part of an extreme programming team of 8 programmers that has been tasked to build a new online platform (initially just a web site) called "**BodyBook**" (or BB for short). The objective of BB is to provide a service very similar to FaceBook, but where the faces of people in pictures or videos, if visible, are automatically obscured/censored by the software which replaces them with black ellipses covering the faces.

You are asked to sketch a release plan for BB. In your release plan please specify:

a) A list between 12 and 15 stories identified for the release. Note: please limit the stories to the end users of the system. That is, do not consider any other users (e.g., system administrators, the company itself, advertisers, etc.). [15%]

b) A priority for each story in terms of value for BB's users, using a point system where 3 = very valuable, 2 = valuable, 1 = optional. [5%]

c) An estimate of the programming effort for the team required by each story based on the following scale: [5%]

E = Easy (takes 0 to 2 working days to the team)

M = Medium (takes 3 to 5 working days to the team)

D = Difficult (2 weeks)

X = eXtremely difficult (3 or more weeks)

Question 4

Clean code is an essential element of extreme programming.

Someone has started developing code to keep track of stocks and orders in a retailer business. The code has the ability to store the products in stock in a simple database (a comma-separated value, or CSV, file) and to perform operations like listing the content of the database or processing orders. The main program at present simply tests these operations. In particular, it processes three orders: (1) an order of two "web browser for children" (product ID: 1732), (2) an order of one "Hand-held compressor" (product ID: 2338), and (3) a further order of *two hundred* "web browser for children" (product ID: 1732), which cannot be processed as it exceeds the available stock.

If file "product_database.csv" contains just the following two lines (representing two products)

```
1732,Web browser for children,10,122.20
2338,Hand-held compressor,2,321.50
```

running the program produces the following textual output:

```
Current Stock
Ref: 1732 Desc: Web browser for children Stock: 10 Price: 122.20
Ref: 2338 Desc: Hand-held compressor Stock: 2 Price: 321.50
Not enough stock for item 1732
Current Stock
Ref: 1732 Desc: Web browser for children Stock: 8 Price: 122.20
Ref: 2338 Desc: Hand-held compressor Stock: 1 Price: 321.50
```

As a result of running the program the content of the file "product_database.csv" becomes:

```
1732,Web browser for children,8,122.20
2338,Hand-held compressor,1,321.50
```

Below you will find three implementations of the code: one in Java, one in Python and one in C. Choose one implementation, study it, and the list the places (use line numbers) and ways in which the code is not clean code. In particular, please consider:

a) Names [6%]

b) Functions/Methods [8%]

c) Comments [6%]

d) Formatting [6%]

e) Data Structures and Classes [4%]

Java Implementation

```java
import java.io.*;

public class Business {
    int numProducts;
    Prdct[] products;

    class Prdct {
        int referenceNumber; // A unique number for the item
        String description; // Space for description
        int k; // An index
        float price;
    }

    boolean buyProduct(int refNum, int quantity) {
        for (int i = 0; i < numProducts; i++)
            if ( products[i].referenceNumber == refNum &&
                    products[i].k >= quantity ) {
                products[i].k -= quantity;
                return true;
            }
        return false;
    }

    void stashAway(String databaseFileName) throws IOException {
        PrintWriter productDatabase=new PrintWriter(new FileWriter(databaseFileName));
        for (int i = 0; i < numProducts; i++)
            productDatabase.format("%d,%s,%d,%.2f\n", // The commas are necessary!
                    products[i].referenceNumber, new String(products[i].description),
                    products[i].k, products[i].price);
        productDatabase.close();
    }

    void printProducts() {
        for (int i = 0; i < numProducts; i++)
            System.out.printf("Ref: %d Desc: %s Stock: %d Price: %.2f\n",
                    products[i].referenceNumber, new String(products[i].description),
                    products[i].k, products[i].price);
    }

    void goGetThem(String databaseFileName) throws FileNotFoundException {
        BufferedReader productDatabase=new BufferedReader(new FileReader(databaseFileName));
        for (int i = 0; i < 100; i++)
            try {
                String record = productDatabase.readLine();
                String elements[] = record.split(",");
                products[i].referenceNumber = Integer.parseInt(elements[0]);
                products[i].description = elements[1];
                products[i].k = Integer.parseInt(elements[2]);
                products[i].price = Float.parseFloat(elements[3]);
            } catch (Exception e) {
                numProducts = i;
                break;
            }
    }

    void run() throws IOException {
        // Printing products
        System.out.println("Current Stock");
        printProducts();
        // Processing customer orders
        if ( !buyProduct(1732,2))
            System.out.println("Not enough stock for item 1732");
        if ( !buyProduct(2338,1))
            System.out.println("Not enough stock for item 2338");
        if ( !buyProduct(1732,200))
            System.out.println("Not enough stock for item 1732");
        // Printing products
        System.out.println("Current Stock");
        printProducts();
        stashAway("product_database.csv");
    }

    Business() throws FileNotFoundException {
        products = new Prdct[100];
        for(int i = 0; i < 100; i++)
            products[i]=new Prdct();
        goGetThem("product_database.csv");
    }

    public static void main(String[] args) throws IOException {
        Business myBusiness = new Business();
        myBusiness.run();
    }
}
```

Python Implementation

```
MAX_NUM_PRODUCTS = 100

class Prdct(object):
    def __init__(self):
        self.reference_number = 0 # A unique number for the item
        self.description = ""
        self.k = 0 # An index
        self.price = 0

class Business(object):
    def buy_product(self, ref_num, quantity):
        for i in range(self.num_products):
            if ( self.products[i].reference_number == ref_num and
                    self.products[i].k >= quantity ):
                self.products[i].k -= quantity
                return True
        return False

    def stash_away(self, database_file_name):
        product_database=open(database_file_name,"w")
        for i in range(self.num_products):
            product_database.write("%d,%s,%d,%10.2f\n" % # The commas are necessary!
                    ( self.products[i].reference_number,
                      self.products[i].description,
                      self.products[i].k,
                      self.products[i].price ))
        product_database.close()

    def print_products(self):
        for i in range(self.num_products):
            print("Ref: %d Desc: %s Stock: %d Price: %10.2f"%
                  (self.products[i].reference_number,
                   self.products[i].description,
                   self.products[i].k,
                   self.products[i].price))

    def go_get_them(self, database_file_name):
        product_database=open(database_file_name,"r")
        for i in range(MAX_NUM_PRODUCTS):
            try:
                record = product_database.readline()[:-1]
                elements = record.split(",")
                self.products[i].reference_number = int(elements[0])
                self.products[i].description = elements[1]
                self.products[i].k = int(elements[2])
                self.products[i].price = float(elements[3])
            except Exception:
                self.num_products = i
                break

    def run(self):
        # Printing products
        print("Current Stock")
        self.print_products()
        # Processing customer orders
        if not self.buy_product(1732,2):
            print("Not enough stock for item 1732")
        if not self.buy_product(2338,1):
            print("Not enough stock for item 2338")
        if not self.buy_product(1732,200):
            print("Not enough stock for item 1732")
        # Printing products
        print("Current Stock")
        self.print_products()
        self.stash_away("product_database.csv")

    def __init__(self):
        self.num_products = None
        self.products = [Prdct() for i in range(100)]
        self.go_get_them("product_database.csv")

if __name__ == "__main__":
        my_business = Business()
        my_business.run()
```

C Implementation

```c
//
// Program written by R. Poli
// version 32
//

// Including the standard header files
#include <stdio.h>
#include <stdlib.h>

typedef struct Prdct {
  int reference_number; // A unique number for the item
  char description[1000]; // Lots of space for description
  int k; // An index
  float price;
} Prdct;

int buy_product(Prdct *products, int num_products, int ref_num, int quantity) {
    int i;
    for (i = 0; i < num_products; i++)
        if ( products[i].reference_number == ref_num &&
                products[i].k >= quantity ) {
            products[i].k -= quantity;
            return 1;
        }
    return 0;
}

void stash_away(Prdct *products, int num_products, char*database_file_name) {
    FILE *product_database=fopen(database_file_name,"w");
    for (int i = 0; i < num_products; i++) {
        fprintf(product_database,
                "%d,%s,%d,%.2f\n", // The commas are necessary!
                products[i].reference_number, products[i].description,
                products[i].k, products[i].price);
    }
    fclose(product_database);
}

int print_products(Prdct *products, int num_products) {
    int i;
    for (i = 0; i < num_products; i++) {
        printf("Ref: %d Desc: %s Stock: %d Price: %.2f\n",
                products[i].reference_number, products[i].description,
                products[i].k, products[i].price);
    }
    return i;
}

int go_get_them(Prdct *products, char*database_file_name) {
    FILE *product_database=fopen(database_file_name,"r");
    int i;
    for (i = 0; i < 100; i++) {
        int items_read = fscanf(product_database, "%d,%[^,],%d,%f",
                &(products[i].reference_number), products[i].description,
                &(products[i].k), &(products[i].price));
        if (items_read != 4)
            break;
    }
    return i;
}

int main(void) {
    Prdct products[100];
    int num_products = go_get_them(products, "product_database.csv");
    // Printing products
    printf("Current Stock\n");
    print_products(products, num_products);
    // Processing customer orders
    if ( !buy_product(products,num_products,1732,2))
        printf("Not enough stock for item 1732\n");
    if ( !buy_product(products,num_products,2338,1))
        printf("Not enough stock for item 2338\n");
    if ( !buy_product(products,num_products,1732,200))
        printf("Not enough stock for item 1732\n");
    // Printing products
    printf("Current Stock\n");
    print_products(products, num_products);
    stash_away(products, num_products, "product_database.csv");
    return 0;
}
```

END OF EXAM PAPER CE320-6-AU/AT

Once you have completed your answers, please upload them to FASER
http://faser.essex.ac.uk

Remember to add your REGISTRATION NUMBER onto ALL documents that you upload.