SEG 3300 – Introduction to Software Engineering

Professor: Robert L. Probert

DEFERRED FINAL EXAMINATION: July 6, 2000

Name: _____

Student Number: _____

Duration: 3 hours

Closed book

Calculator allowed

Part 1: /40 Part 3: /20

Part 2: /25 Part 4: /15

Total: /100

Part 1/40 marks (2 marks per blank)

Fill in the blanks with the MOST APPROPRIATE answer (one word or phrase per question).

Question 1.1

In the typical software life cycle, _____ is the most expensive phase.

Question 1.2

The best times in software development to find faults in a new system are during

_____ and _____.

Question 1.3

The two most useful diagrams in object–oriented analysis and design are

_____ and _____.

Question 1.4

A piece of software hurriedly put together that incorporates much of the important functionality of the intended product, but omits detailed aspects invisible to the user is called _____

_____.

Question 1.5

A specification such as "If p.essure is greater than 32 psi, shut the value, and if the pressure exceeds 32 psi, alert the operator and wait for the operator to shut the valve" is best described as

_____.

Question 1.6

Product Specifications are intended to describe

_____,

whereas product design is intended to describe

_____.

Question 1.7

In the case of design. traceability means

_____.

Question 1.8

The final phase of testing in namely acceptance testing, has the following specific characteristics:

_____

_____

_____

Question 1.9

The term _____ refers to the degree of interaction between modules or components.

Question 1.10

A use case diagram shows _____ and

_____.

Question 1.11

Data flow analysis is a design technique for achieving modules and

components with high

_____.

Question 1.12

Function Points (FP) are superior to Lines of Code (LOC) in Project Cost Estimation because:

_____
_____.

Question 1.13

One of the improvements for Cost and Effort Estimation in COCOMO II over COCOMO is

_____.

Question 1.14

The main goal of testing is to

_____.

Question 1.15

If two modules have read/write access to the same global variable, we say the modules are

_____–coupled.

Part 2/25 marks

A point–of–sale terminal (POST) records all the items purchased by a customer, presents a sale total, collects a payment, and produces an itemized receipt for the customer. The Cashier scans the universal product code (UPC) from each item. If there is more than one of the same item, the cashier can enter a quantity as well. The POST determines the item price and adds the item information to the running sales transaction. This description and price of the current item are displayed. Only a manager can start up or shut down the POST system. A system administrator can add new users (cashiers) to the system.

Underline the candidate classes and circle the attributes of classes in the aboye description. (5 marks)

Give a significant secondary scenario for the POST system. Explain why this is a secondary scenario. (5 marks)

Interaction Diagrams may be either $equence diagrams or collaboration diagrams. Although these two types of diagrams are equivalent, each has a different emphasis. State the emphasis for each type of diagram:

Sequence Diagrams emphasize

_____
_____
_____

Collaboration Diagrams emphasize

_____
_____
_____

Given the following diagram, state the type of diagram it is, and draw its equivalent in the other type of diagram.

makePayment(cash)

1:makePayment(cash)

1.1:create(cash)

d) Draw a State Diagram to represent the operation of the POST system for one customer. Assume the initial state has no customers, but has been turned on, and is being attended by a cashier. State any other assumptions you feel are necessary. (5 marks)

e) Complete the following class diagram cardinalities on the associations: (5 marks)

Part 3/20 marks

In this question, you are asked to develop test cases based on code coverage.

The following is pseudo–code for part of the elevator controller and an associated control flow–graph. The pseudo–code describes what to do according to whether an elevator button has been pressed, whether the elevator is stopped or moving, and whether the upcoming floor button has been pressed.

Note: only PART of the code and PART of the flowgraph is given.

Partial Pseudo–code for Elevator Controller

do forever

```
{
    if ( a button has just been pressed)
        if (button is not on)
        {
            button.turn button on;
            log request;
        }
    else if (elevator is moving up)
    {
        if (there is no request to stop at floor f)
            elevator.move one floor up;
        else
        {
            stop elevator by not sending a message to move;
            elevator doors.open doors;
            if (elevator button is on)
                elevator button.turn button off;
            update requests;
        }
    }

    else
        there are no requests, elevator is stopped with elevator doors
closed, so do nothing;
}
```

Example Test Case

TEST CASE SET UP:

Elevator is moving up with its doors closed and has just passed Floor 2. No buttons are on.

TEST CASE:

User presses Up button at floor 3.

Verify:

Up floor button at floor 3 is turned on.

Verify:

Elevator stops at floor 3.

Verify:

Up floor button at floor 3 is turned off.

Verify:

Elevator doors open.

Verify requests are updated.

VERDICT:

System PASSES this test if no verify step is failed.

What branches are covered in the flowgraph by this test case? Fill in the table below with an X wherever a branch is executed at least once by this test case. (7 marks)

Predicate Label	TRUE Branch	FALSE Branch
P1	X	
P2		
P3		
P4		
P5		

Give a test case to cover the branch P4: TRUE.

TEST CASE SET UP: (where the elevator is now, whether it is stopped or which direction it is going in, what floor buttons are on (if any), what elevator buttons are on (if any), and whether the doors are open or closed), (9 marks)

TEST CASE:

VERDICT

System passes this test if:

What are the advantages and disadvantages (2 each) of ensuring that all branches in the code are exercised by some test case? (4 marks)

Part 4/15 marks

You are asked to plan for the last stages of a small software project (implementation, integration and product testing). The architectural design of the software produced the following decomposition

(sometimes called a module invocation hierarchy).

The implementation and integration is done by 2 developers (Eric and Marc) and the number of days needed to implement and integrate each module is as follows:

Activity	Developer	Days
1. Implement & Integrate A	Eric	3
2. Implement & Integrate B	Eric	2
3. Implement & Integrate C	Marc	2
4. Implement & Integrate D	Eric	4
5. Implement & Integrate E	Marc	3
6. Implement & Integrate F	Marc	5
7. Implement & Integrate G	Eric	2
8. Product Test		5

NOTE: The activity Number IN NO WAY reflects necessarily the order of activities. The order of activities is determined by the integration strategy. Note that the same person cannot work on 2 modules at the same time.

Note: you must find out the tasks and dependencies in order to answer the following questions.

Also, state and justify any assumptions you feel are necessary.

Assume a BOTTOM–UP implementation and integration approach.

a) Draw a PERT diagram showing the order and estimated duration of implementation, integration and product testing activities. (5 marks)

b) What is the minimal duration of the project? (1 mark)

c) Give the earliest, latest and slack of each task by filling in the table below. (7 marks)

Activity (task) Number	Developer	Earliest	Latest	Slack

d) What are the critical path activities? (2 marks)