香 港 中 文 大 學
The Chinese University of Hong Kong

Course Examinations 2nd Term, 2011-2012

Course Code & Title : CSCI 3100 Software Engineering

Time allowed : 3 hours 0 minutes

Student I.D. No. : Seat No. :

Part I

All **Part I** questions are multiple choice questions. Each multiple choice question has **exactly one** correct answer. Each question is worth 1.5 points. An incorrect answer receives -0.5 points (negative points), and an unanswered question receives 0 points. Put down the corresponding answer in the following boxes. Answers elsewhere will not be marked. The maximum score of **Part I** is 60 points.

Multiple-Choice questions not provided

P.1 – P.12

Please tick the answer (mark with "√") in the following boxes provided.

Question	A	B	C	D	E
1.					
2.					
3.					
4.					
5.					
6.					
7.					
8.					
9.					
10.					
11.					
12.					
13.					
14.					
15.					
16.					
17.					
18.					
19.					
20.					

Question	A	B	C	D	E
21.					
22.					
23.					
24.					
25.					
26.					
27.					
28.					
29.					
30.					
31.					
32.					
33.					
34.					
35.					
36.					
37.					
38.					
39.					
40.					

Part II

NOTE:

(1) All **Part II** questions (totally six questions) are essay questions, although most expected answers are short and precise. The maximum score of **Part II** is 60 points.

(2) Write **all** your Part II answers on the Answer Book.

Question 1: Specification by Data Flow Diagrams (10 Points)

In the hope of having a real democratic Chief Executive Election and universal suffrage by 2017, in a high-level meeting with several government officers, a discussion on the voting system for Chief Executive Election was held in order to pave the way for free election.

Due to the Executive Council's confidentiality rule, only you, as the secretary of the meeting, are allowed to jot down the details discussed in the meeting. However, your boss, Leung, is afraid of what he has said in the meeting will be improperly disclosed by his opponent Tang in the future. He asked you to describe the following conversation about a **voting system** at a polling station as detail as possible by using a **data flow diagram**:

Leung: We should try our best to achieve real democratic Chief Executive Election by 2017!

Tang: Yes, this is the core value of Hong Kong.

You: Um... so how exactly do we vote for the Chief Executive in 2017?

Tang: Let's make it fully automated by a computer system. A "voter" input his/her "choice of candidate" into the "voting" system. After that, a "ballot" will be generated and stored in a "ballot database". When we have to "count" the ballots, we just get the "ballot" from the "ballot database" and display the "polling result" on the "screen" in the polling station.

Leung: That sounds solid.

You: I am afraid that this system is too simple. Shouldn't we at least perform some verification on the voters? Otherwise, maybe tourists from other places can vote for the Chief Executive as well!

Leung (smiling face): Yeah! Of course, only legitimate Hong Kong residents with qualification can vote. We have a "qualification database" which serves for this purpose.

You: So the "voter" should first present his/her HKID card to the smart card reader. The "HKID" together with "voter detail" from the "qualification database" would be used to "verify the qualification" of the voter.

Tang: Then the "verified ID" and the "choice of candidate" from the voter would carry out the "voting" procedure that I have just described.

Leung (smiling even more): We should notify the voter that they have voted me... No, I mean, their "choice of candidate" on a success vote.

You: ..... For security reason, I think that before we "count" the "ballot", we should "verify the ballot" with "voter detail" from the "qualification database" once more. Then we can safely "count" the "verified ballot" in that polling station.

Leung: So be it! Thanks everyone for your great ideas! I will keep listening to the requests of Hong Kong residents and become a popular Hong Kong Chief Executive!

Precisely describe the agreed 2017 Chief Executive Election **voting system** using a **data flow diagram**. Note all the keywords quoted in the meeting conversation should be properly captured in your DFD.

Question 2: Specification by Finite State Machines (10 Points)

Finite state machines are often used to process character strings such as binary integers (e.g., "100" represents "4" and "011" represents "3"). In this problem we are going to use FSM to process the mod operation for a binary integer.

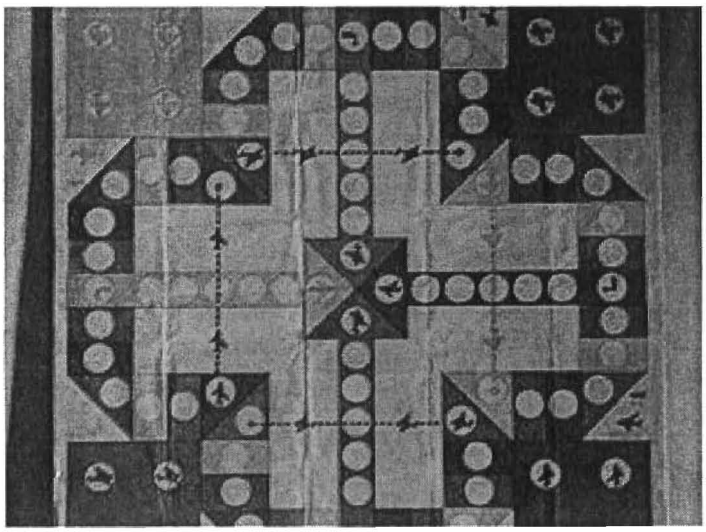
(1) Given a binary integer, design a finite state machine that read it from "left to right", i.e., most significant bit first, and compute its remainder mod 2. For example, for the input "100" the resulting FSM should be in state "0" and for the input "011" the resulting FSM should be in state "1". Note the input can be an arbitrarily long binary integer. The FSM should use minimal states, and it does not produce any output. Mark the initial state properly. (3 points)

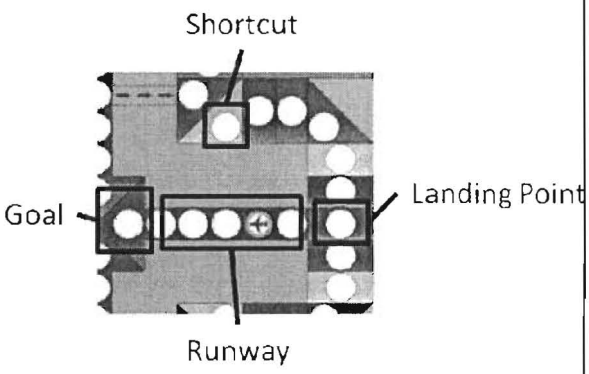
(2) Given a binary integer, design a finite state machine that read it from "left to right", i.e., most significant bit first, and compute its remainder mod 3. Other conditions are the same as in (1). (3 ponts)

(3) Given a binary integer, design a finite state machine that read it from "right to left", i.e., least significant bit first, and compute its remainder mod 3. Other conditions are the same as in (1). (Hint: A difficult problem. You need 6 states) (4 points)

Question 3: Software Requirement and Design with UML (10 points)

The "aeroplane chess game" is a very classic Chinese board game. In this game, there are 4 "players", each of whom has 4 "chesses" of his/her own color. The goal of this game is to have all of the 4 chesses taking off, completing a cycle in clockwise direction, and arrive at the goal in the middle as quick as possible. There are a total of 76 "squares" in the "chess board". Some squares behave differently when a chess is stopping or moving on them. For example, the "shortcut", "landing point", "runway" and the "goal" are these squares. Each player takes turn to roll a "dice", picks a chess of his/her choice, and moves accordingly. You are asked to implement this game using objected oriented design.





(1) Based on the above information, draw a UML class diagram showing the association and inheritance relationship between different classes (the classes are quoted in the question). You don't need to draw the data and method of each class. (5 pints)

(2) On a player's turn, the "player" call the "roll()" function of the "dice" object. Upon knowing the "dice value", the player picks a chess and calls the "moveChess(chess, dice value)" function of the "chess board" object. The "chess board" then call the "update()" method of the "chess" object and the "square" object to refresh their status. Draw a UML sequence diagram for the above scenario. (5 points)

Question 4: Software Design with TDN and Stepwise Refinement (10 points)

In this program we would like to design a simple resource management system. Suppose we have a reference book for CSCI3100 course, we would like to lend it to students. (Students are identified by their unique student ID.) Since we have only one book, we can lend it to **at most one** student at the same time. There is a waiting list of **N** users. If there are more than N requests, we would accept the first N requests only and reject the remaining. The students who have been rejected could apply for borrowing the book again later on. The waiting list is in a **first in first out order,** which means the user who required the book earlier should get the book earlier. To make it fair, a student can borrow the book **only one time**. If a student has already borrowed the book, he/she **cannot** request it again (the procedure would reject this kind of request). Moreover, a student can keep the book for **at most** two days. That is, He/She should return the book within two days, otherwise, the management system would recall the book. (You can assume the recall is always successful). Once the book is returned, it would be given to the first student in the waiting list.

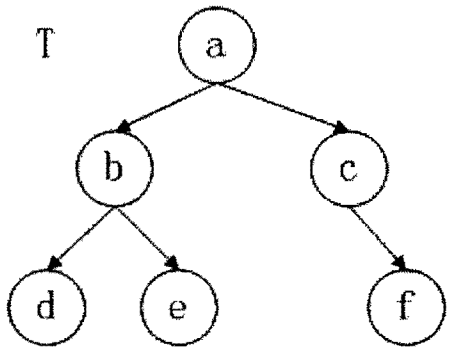
(1) Using TDN, describe design of the resource management system. Concentrate your design in the *module interface*. The *implementation* section could be made short by using comments. (5 points)

(2) Now use stepwise refinement to describe the implementation of the procedure for ordering the book. The waiting list is a **static array of length N**. You can do this for just two iterations. First provide a high level (but non-trivial) design of the implementation of the spelling checker, and then refine the design in more detail. If the module needs to use other modules, identify them clearly. (5 points)

Question 5: Software Design and Implementation (10 points)

A tree is a widely-used data structure that emulates a hierarchical tree structure with a set of linked nodes. The following figure shows a typical tree structure.

Height is one of the basic properties of a tree. It is important in many tree applications, e.g., search trees. The height of a node is the length of the **longest** downward path to a leaf from that node. The height of the root is the height of the tree. For example the height of the tree in the following figure is 3. (Note: Height of node *f* is 1.)



Let's consider the binary tree defined as follows:

```
public class TreeNode {
      TreeNode left;
      TreeNode right;
      Object value;

      TreeNode(Object v){this.value = v;}
      TreeNode(Object v,TreeNode left,TreeNode right)
      {
            this.value = v;
            this.left = left;
            this.right = right;
      }
}

public class Tree {
      private TreeNode root;
      public Tree(TreeNode r){this.root = r;}
}
```

To compute the height of the tree, we compute the height of root. The height of a tree node is computed by 1 + the biggest height of its children.

(1) The following function is a recursive solution. Fill out the 6 numbered blanks in the following function and put them in the Answer Book with the corresponding numbers. What is the time complexity of this solution? (5 points)

```
public class Tree {
      ...
      //this function computes the height of the tree
      public int GetHeight_recursive()
      {
            return _____①_____;
      }

      //this function computes recursively the height of a node
```

```
    private int GetHeight_recursive(TreeNode t)
    {
        if(t== null)
            return 0;

        int height = 1;
        int lheight = 0,rheight = 0;

        if(t.left != null)
            lheight = _____②_____;
        if(t.right != null)
            rheight = _____③_____;

        height += ____④____? ___⑤___ : ____⑥___;
        return height;
    }
}
```

(2) The following function is a non-recursive solution. This function uses the trick that the height of a tree equals to the **maximum depth** of the tree. The **depth** of a node is the length of the path to its root (i.e., its root path). For example, in the example figure, the depth of root ***a*** is 1, the depth of node ***f*** is 3. Fill out the 7 numbered blanks in the following function and put them in the Answer Book with the corresponding numbers. What is the time complexity of this function? (5 points)

```
public class Tree {
    …
    //this function computes the maximum depth of the tree
public int GetHeight_nonrecursive() {
        if (root == null)
            return 0;
        //the stack to move around the tree nodes
        Stack<TreeNode> nodeStack = new Stack<TreeNode>();

        //the stack to keep the depth of the nodes in related position of
        node stack
        Stack<Integer> depthStack = new Stack<Integer>();

        nodeStack.push(root);
        depthStack.push(1);     // push the root depth

        int maximumDepth = 0;
        int depth;

        while (nodeStack.isEmpty() == ____①____) {
            TreeNode node = nodeStack.pop();
            depth = depthStack.pop();
            if (_____②_____) {
                _____③_____;
            }

            if (node.left != null) {
                _____④_____;
                _____⑤_____;
            }
            if (node.right != null) {
                _____⑥_____;
                _____⑦_____;
            }
        }
        return maximumDepth;
    }
}
```

Question 6: Software Testing and Verification (10 points)

The following search procedure takes an element table of ordered elements and a key element, and find if the key can be found in the table.

```
procedure search(key: in element; table: in element Table; found: out Boolean)
begin
      bottom := table'first;  top := table'last;
      while bottom < top loop
            if (bottom + top) rem 2 ≠ 0 then
                  middle := (bottom + top - 1) /2;
            else
                  middle := (bottom + top)/2;
            end if;
            if key ≤ table(middle) then
                  top := middle;
            else
                  bottom := middle + 1;
            end if;
      end loop;
      found := key = table(middle);
end search;
```

(1) The element table contains ***n*** elements. Write the pre- and post-condition using formal logic to specify the property of this program. What is the name of the search procedure (3 points)

(2) Draw the control flow graph of this program. In this control flow graph, calculate the McCabe's number, or V (G). Design a set of test cases to achieve 100% path coverage (i.e., C_3). You only need to specify the input. (4 points)

(3) Design a test set to perform black-box testing for the program. You only need to state clearly your testing strategy. Inputs of your design are not required. Note there is a bug in the given procedure. State the bug and how to fix it. (**3** points)

-End-

Have a great summer break!