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

Course Examinations 2nd Term, 2013-14

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 points (no 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. 2 to 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 Diagram (10 points)

Alice and Bob, students of CSCI3100, are working on their project. Alice is in charge of the security of the project website. She is concerned with the news that NSA has hacked into a lot of websites and stolen the username password combination of millions of users. She wants to make sure that their site is secure so that even if NSA hacked into their site, they still cannot recover and steal the passwords. So she consulted Bob, who is a security expert. The followings were their conversation.

Alice: Hi Bob, do you know that NSA is spying on a lot of websites? I heard that they recovered a lot of passwords! That's awful. I use the same password for all my websites!

Bob: Yes. Those websites are irresponsible for their users. They store a user's password as md5 hash with no salt. There exists a million ways to recover an unsalted password!

Alice: But why? I was taught that security hash function such as md5 should be very hard to reverse.

Bob: Yes that's true theoretically if you use a completely random long password. But most passwords we use are words that can be found in a dictionary. NSA can pre-compute the md5 of all the words and simple combinations of all the words. Then given an md5, they can look up the table to find the corresponding password. This is called dictionary attack. Even worse, they can pre-compute the md5 hash for all the possible passwords up to certain length, for example 8. Then all possible passwords of length under 8 can be recovered instantly by looking up the pre-computed table!

Alice: Hmm, I see. I want to make sure our website is secure. I heard you mention password salt. What is that? Can you talk more about that?

Bob: Sure. In fact the idea of salted password is quite simple. When a registrant is registering for the website, he/she provides the username and password. Our site should generate a long random string called salt. Then the password and the salt are exclusive-ored (XOR operation) to get a salted password. Apply md5 on this salted password and you get the hashed value. Then register function store the hashed value as well as the username in the login database, store the salt and the username in the salt database.

Alice: But why would this salt help preventing NSA from getting the original password?

Bob: The salt is long and randomly generated. Also it should be different for every user. When NSA wants to brute force attack md5 hash value of the salted password, without the salt, they have to try all possible combinations as long as the salt. Using a salt of 2048 bit for example, would require millions of years of computation, rendering the brute force hack impossible! Even if they hacked into the salt database, they still need to brute force attack all possible passwords and this cannot be pre-computed. This makes a properly salted password very hard to recover.

Alice: That sounds great! But how do you authenticate the user?

Bob: That's easy! When the user is trying to login, he or she should provide the username-password combination. You retrieve the user's salt and hash value from salt database and login database respectively using the username. You XOR the salt with the password to get the salted password. Then apply md5 hash to get the hash value and compare it with the hash value retrieved. Display the results to the screen then you can authenticate the user.

Alice: Thanks for the explanation! I'm going to implement the most secure website in the world!

Based on their conversation, draw the Data Flow Diagram of the registration and authentication process described.

Question 2: Finite State Machine and Petri Net (10 points)

After tiresome work for CSCI3100 project, Alice and Bob are often playing a special game that is similar to the classic Tic-Tac-Toe game. To play this game, they take turns marking the spaces in a 1x7 grid in any unmarked space. Alice marks 'X' and Bob marks 'O'. Alice always goes first. Different from traditional tic-tac-toe game, in this game, one player wins once the other guy places three adjacent marks. For example, in the grid shown below, as Alice places three adjacent 'X' marks, Bob wins in this case according to our definition. If nobody wins and 7 marks are already placed, the game comes to a draw.

O	X	X	X	O	X	O

(a) If we use a Finite State Machine to specify the behaviour of this game, how many *possible* states are needed to describe different situations regarding 1 space, 2 spaces and 3 spaces are marked respectively? You are not required to draw the diagram. You just need to write down 3 numbers for these 3 cases. (3 points)

(b) Based on the answer of (a), what problem of finite state machine is revealed when we use it in software engineering? (2 points)

(c) Please draw a Petri Net to model this problem. Hints: Your places should contain "Alice Wins", "Bob Wins", "Draw" and some other places. Please handle the "Draw" condition carefully. Remember to draw your initial tokens. (5 points)

Question 3: Software Design and UML (10 points)

In this problem, you are required to design a web service for patients who want to make appointments with doctors through the Internet. For simplicity, we only consider one function: send appointment requests from patients to doctors.

The procedure of this function is introduced as follows: a *patient user* uploads an appointment request to the *server*, and the *server* stores it into the *database* and sends notification email to inform the *doctor user* that he/she has a new request. Later, when a *doctor user* wants to check new requests, he/she sends an HTTP request to the *server* to load the web pages. Before the *server* replies, it asks the *database* for the previous appointment requests data. Once the *server* receives the data, it sends the data to the *doctor user* with HTTP reply. The whole process completes the transmission of appointment request from the patient users to the doctor users by using a stateless web server and a database.

(a) Please draw a UML sequence diagram to show this procedure that involves the following entities: *patient user*, *server*, *database* and *doctor user*. (5 points)

To implement this web service, you are suggested to use object-oriented design in your implementation for higher maintainability. You need to consider two main classes: *User* and *Text*. For the *User* class, two subclasses are considered: *PatientUser* and *DoctorUser*. For the *Text* class, it also has two subclasses *AppointmentText* and *NotificationText*, which are corresponding to the appointment requests and notification emails respectively.

(b) Please draw a UML class diagram illustrate the relationship between these 6 classes. Hints: use only inheritance and association relations. (5 points)

Question 4: Stepwise Refinement (10 points)

Counting sort is a sorting technique based on keys between a known range. It works by counting the number of objects having distinct key values. It then performs some arithmetic to calculate the position of each object in the output sequence.

Let us understand it with the help of an example:

Consider the data in the range 0 to 9.

Input data: 1, 4, 1, 2, 7, 5, 2

1) Take a count array to store the count of each unique object.

Index: 0 1 2 3 4 5 6 7 8 9

Count: 0 2 2 0 1 1 0 1 0 0

2) Modify the count array such that each element at each index stores the accumulate sum of previous counts. The modified count array indicates the starting position of next object in the output sequence.

Index: 0 1 2 3 4 5 6 7 8 9

Count: 0 2 4 4 5 6 6 7 7 7

3) Output each object from the input sequence followed by decreasing its count by 1.

Process the input data: 1, 4, 1, 2, 7, 5, 2. count of 1 is 2.

Put data 1 at index 2 -1 = 1 (*notice: array index start from 0*) in output. Decrease count by 1 since we have gone through one data already. If next data 1 is found it would be placed at an index 0.

Question: Please (1) *fill in the blanks* with nature language (English) to describe what the content is and (2) *draw the corresponding decomposition tree* to demonstrate this stepwise refinement process.

Step 1:

let **n** be the length of the input array **a** to be sorted, **RANGE** be the range of values of array **a**, **count** be the counting array described in the example, and **b** be the array after sorted
count appearance of each value
calculate starting position of next value
fill in output array

Step 2:

let **n** be the length of the input array **a** to be sorted, **RANGE** be the range of values of array **a**, **count** be the counting array described in the example, and **b** be the array after sorted

```
i := 0;
_____①_____
while i < n loop
        _____②_____
end loop;
i := 1
while i < n loop
        _____③_____
end loop;
i := 0;
while i < n loop
        _____④_____
        _____⑤_____
end loop;
```

Step 3:

let **n** be the length of the input array **a** to be sorted, **RANGE** be the range of values of array **a**, **count** be the counting array described in the example, and **b** be the array after sorted

```
i := 0;
_____①_____
while i < n loop
        count(a(i)) := count(a(i)) + 1
end loop;
i := 1
while i < n loop
        count(i) := count(i) + count(i-1)
end loop;
i := 0;
while i < n loop
        b(count(a(i) – 1)) := a(i)
        count(a(i)) := count(a(i)) – 1
end loop;
```

Question 5: Software Design and Implementation (10 points)

There are many ways to **multiply two positive integer numbers**. One interesting method is the Russian peasant algorithm. The idea is to double the first number and halve the second number repeatedly till the second number becomes less than 1. In the process, whenever the second number become odd, we add the first number to result (result is initialized as 0). The idea behind it is the equation: $a*b = (a*2) * (b/2)$ if b is even, or $a*b = ((a*2)*(b/2) + a)$ if b is odd. If we keep on applying this equation, we could reduce b and increase a until we get b to be less than 1.

Let's see an example of 238*13 = 3094:

238	13
~~476~~	~~6~~
952	3
+1904	1
N/A	0
3094	

We keep on doubling the left and halving the right. Lines with even numbers on the right column are struck out, and the remaining numbers on the left are added, giving the answer as 3094.

Question: Please *fill in the blanks* of *both* the recursive version and non-recursive version of the algorithm below. The algorithm takes *only positive integers* as input, and you can assume the result *will not exceed* the maximum of integer representation.

Requirement: One advantage of this algorithm is that it does not rely on multiplication table. You could use only the bitwise operations and add operation to implement it. To fill the blanks, you are ***only allowed*** to use operations including: & (bitwise and), | (bitwise or), ! (not), << (left shift), >> (right shift), +, -, >, >=, <, <=, = =, =. All other operations like: *, /, % (compute the reminder of divide), etc. are *not allowed!*

Recursive algorithm:

```
int russianMulti(int a, int b)
{
  if (a <= 0 || b <= 0) return -1;
  return russianMultiRecursive(a, b, res);
}

int russianMultiRecursive(int a, int b, int res)
{
  if (_____①_____)
  {
    if (_____②_____)
    {
      _____③_____
    }
    _____④_____ = russianMultiRecursive(__⑤__,__⑥__, res);
  }
  return res;
}
```

Non-recursive algorithm:

```
int russianMulti(int a, int b)
{
  if (a <= 0 || b <= 0) return -1;
  int res = _____⑦_____;
  while (_____⑧_____)
  {
       if (_____⑨_____)
            _____⑩_____;
       _____⑪_____;
       _____⑫_____;
   }
   return res;
}
```

Question 6: Software Testing and Verification (10 points)

Given the following Java program:

```
1.      public int Transform(String s) throws Exception{
2.        int result = 0;
3.        int power = 0;
4.        for (int i = s.length() - 1; i >= 1; i--){
5.            if(s.charAt(i) == '1')
6.                 result += Math.pow(2,power);
7.            else if(s.charAt(i) != '0')
8.                    throw new Exception();
9.            power ++;
10.           }
11.       if( s.charAt(i) == '0')
12.           result = -result;
13.       if( result>2147483648 || result<-2147483648)
14.           throw new Exception();
15.       return result;
16.     }
```

(1) Draw the control flow graph of this program for *edge coverage criterion* (i.e., C_1). (You can lump sequential statements into one node. You don't need to further analyze the external functions.) In this control flow graph, calculate the McCabe's number, or V (G). (3 points)

(2) Design a test set to achieve 100% *statement coverage* (i.e., C_0). Then design additional tests to achieve 100% *edge coverage* (i.e., C_1). Then design additional tests to achieve 100% *condition coverage* (i.e., C_2). Finally design tests to achieve 100% *path coverage* (i.e., C_3). (You only need to specify the input values of each test.) (4 points)

(3) What is does this procedure "Transform" do? Design a test set to perform black-box testing for the program. State clearly your testing strategy and corresponding test inputs. There's a little bug in the program that may throw an exception. Please point out that bug and generate a string that incurs the exception.(3 points)

-End-

Have a great summer break!