104 - Introduction to Computer Science - Quiz four

Name：_____ Student ID：_____

1. What is the difference between **pass-by-value** and **pass-by-reference**? (5%)
   Pass by value means use a copy of the original value stored in memory, i.e., allocate some more memory, copy the value in there and use that memory for the duration of the function. Thus, the original variable isn't affected by the function. Pass by reference means use the actual memory storing the value, so if the function changes the value then the change is immediate and preserved when the function exits. Passing by reference usually leads to faster program execution.
2. Distinguish between **coupling** and **cohesion**. (5%)
   Cohesion is a measure of how closely the processes in a program are related. Coupling is a measure of how tightly two modules are bound to each other.
3. Distinguish between **glass-box** testing and **black-box** testing. (5%)
   Glass-box testing is the testing of all of the functionality of a system and is the responsibility of the programmer. Black-box testing is the testing of a system with no knowledge of the program internal and is done by the system test engineer and the user.
4. According to the following program and the idea of **basis path testing**, determine the basis set of independent paths in a part of the following program. (10%)

```
void delete_element (int value, int array_size, int array[])
{
(1)   int i;
      location = array_size + 1;
(2)   for (i = 1 to array_szie) {
(3)       if (array[i]==value){
(4)           location = i; }}
(5)   for (i = location to array_size) {
(6)       array[i] = array[i+1];
      }
(7)   array_size--;
}
```

   Path 1: 1 – 2 – 5 – 7
   Path 2: 1 – 2 – 5 – 6 – 7
   Path 3: 1 – 2 – 3 – 2 – 5 – 6 – 7
   Path 4: 1 – 2 – 3 – 4 – 2 – 5 – 6 – 7
5. The input data to a program is made up of a combination of three integers in the range 1000 to 1999 (inclusive). A random number generator creates a number between 0 and 0.999. How can this random number generator be used to do random testing for this program? (10%)

For each test we use the following steps:

First, we generate a number between 0 and 0.999,

Second, we multiply the result of part a by 1000 (scaling) to create a number between 0 to 999.

Finally, we add 1000 to the previous result (shifting).

The testing value is between 1000 to 1999.

6. Compare and contrast a **procedural** paradigm with an **object-oriented** paradigm. Describe what the difference between **class** and **method** in an object-oriented language. What is the relation between these two concepts and the concept of an object? (10%)

   In the procedural paradigm, a program is an active agent that manipulates passive objects (data). In an object-oriented paradigm, data are designed as active objects. The actions to be performed on these objects are included in the object.

   In the object-oriented paradigm, a class is the definition of a set of objects. In these languages, a method is an action that can be performed by an instance of the class (an object).

7. We know an array is a sequenced collection of elements, and a linked list is a collection of data too. When do we use the array to storage data in programming? Why don't we use the linked list? Please describe and explain them. (5%)

   An array is used to store the same data type and the elements in the array as the first element, the second element, and so forth until we get to the last element. Each element in linked list contains two parts: data and link. The data part holds the value information. The link is used to chain the data together that contains a pointer (an address) to identify the next element in the list. If we want to store the same type data, we will use the array. If we want to increase access speed, we will use the linked list. Because the linked list has an advantage: insertion and deletion is much easier. But it is not sequential and needs an extra field to record the address of the next mode in memory.

8. What does the following code segment do? Show the result. (10%)

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    int i, j, size = 5;
    for (i = 1; i <= size; i++) {
        for (j = 1; j <= size - i; j++) {
            printf(" ");
        }
        for (j = 1; j <= 2 * i - 1; j++) {
            printf("*");
        }
        printf("\n");
    }
}
```

```
    *
   ***
  *****
 *******
*********
```

9. Find how many **times** the main statement within for loop is executed in the following program and write the code in following program using **recursion function** with C language. (10%)

```
int fundesign(int n){
    int temp=1, x=1;
    for(x = 1; x <= n; x++){
        temp *= x;
    }
    return temp;
}
```

The main statement is executed n times.

```
int fundesign(int n) {
    if (n==1)
        return 1;
    else
        return n* fundesign(n-1);
}
```

10. Write an algorithm that **reverses the elements** of an array so that the last element becomes the first, the second to the last becomes the second, and so forth. (10%)

Algorithm P11-2 shows the routine in pseudocode that reverses the elements of an array.

Algorithm P11-2 *Reversing elements of an array*

```
Algorithm: ReverseArray (A, n)
Purpose: Reverse the elements of an array
Pre: Arrays A with n elements
Post: Array A with elements reversed
Return:
{
    i ← 1
    j ← n
    while (i < j)
    {
        Temp ← A[j]
        A[j] ← A[i]
        A[i] ← Temp
        i ← i + 1
        j ← j – 1
    }
}
```

11. Write an algorithm to **insert** an element into a sorted array. The algorithm must call a search algorithm to find the location for insertion. (10%)

The algorithm that insert an element in a sorted array has two parts. Part a shows the main algorithm. Part b shows the algorithm named shiftdown called by the insert algorithm.

a. Algorithm P11-6a shows the main algorithm.

Algorithm P11-6a *Algorithm for inserting in a sorted array*

```
Algorithm: InsertSortedArray (A, n, x)
Purpose: Insert an element in a sorted array
Pre: A, n, x                          // x is the value we want to insert
Post: None
Return: A
{
    {flag, i} ← BinarySearch (A, n, x)      // Call binary search algorithm
    if (flag = true)                        // x is already in A
    {
        print (x is already in the array)
        return
    }
    ShiftDown (n, A, i)                     // Call shift down algorithm (see below)
    A[i] ← x
    return
}
```

b. Algorithm11-6b shows the auxiliary algorithm used by the main algorithm.

Algorithm P11-6b *The shift-down algorithm used by the insert algorithm*

```
Shiftdown (n, A, i)
Purpose: Shift all elements starting from index i one position down
Pre: A, n, x                          // x is the value we want to insert
Post: None
Return: A
{
    j ← n
    while (j > i − 1)
    {
        A[j + 1] ← A[j]
        j ← j − 1
    }
}
```

12. Write an algorithm to **merge** two sorted linked lists and return the new one as a result. The algorithm must consider the situation that these linked lists may be null. (10%)

```
Node MergeLists(Node list1, Node list2) {
    if (list1 == null) return list2;
    if (list2 == null) return list1;

    if (list1.data < list2.data) {
       list1.next = MergeLists(list1.next, list2);
       return list1;
    } else {
       list2.next = MergeLists(list2.next, list1);
       return list2;
    }
}
```

Reference

http://stackoverflow.com/questions/10707352/interview-merging-two-sorted-singly-linked-list

http://algorithmsandme.in/2013/10/merge-two-sorted-linked-lists-in-one-list/

http://ithelp.ithome.com.tw/question/10161706