UNIVERSITY OF TORONTO
FACULTY OF APPLIED SCIENCE AND ENGINEERING

APS105F — Computer Fundamentals
Final Examination — December, 2004

Examiners: Baochun Li, James MacLean, Andreas Veneris

Duration: 2.5 hours

- Examination Type A: This is a "closed book" examination; no aids are permitted.
- Calculator Type 4: No electronic or mechanical computing devices are permitted.
- Write your answers in the spaces provided. Please answer using a pen or dark pencil.
- If more space is required, blank pages are provided at the back of the examination.
- Rough work, if necessary, can be done on the backs of the pages.
- This examination has 17 pages (including the cover page).
- You must use the Java programming language to answer programming questions.
- You may use methods from the `Math` and `String` classes unless otherwise stated.
- You may assume that the following methods of the class `In` are available:

  `getInt`, `getLong`, `getFloat`, `getDouble`, `getString`, and `getChar`.

- Do not detach any pages from this exam.
- Enjoy the holiday season!

Name _____

Student Number _____ ECF Login _____

MARKS

Question	1	2	3	4	5	6	7	8	Total
Value	10	10	15	20	15	10	10	20	110
Mark									

1. [10 Marks]

For each of the following questions there is only one correct answer. Circle your answer clearly. If we cannot understand what you wrote, you will not receive any credit (1 mark each):

1. **True** or **False**: The declaration `int[][][] c = new int[1][][]` is valid.

2. **True** or **False**: Like instance fields, class fields are initialized automatically.

3. **True** or **False**: A class that includes at least one explicitly defined constructor and has all of its methods declared `private` can never be instantiated.

4. **True** or **False**: A queue ADT is a LIFO ADT.

5. **True** or **False**: Quicksort's performance is always $O(n \log n)$.

6. **True** or **False**: Searching in an array always takes $O(n)$ time.

7. **True** or **False**: Quicksort is suitable for sorting arrays.

8. **True** or **False**: It is always an error to access the `null` character, `(char)0`.

9. **True** or **False**: Storing a type `int` integer requires 32 bits.

10. **True** or **False**: Assume `int [] array` has been properly initialized with n integers. Does the following Java code sort the elements of `array` in ascending order?

```
for (int j = 0; j < n; j++)
  for (int k = 0; k < j; k++)
    for (int l = 0; l < j; l++)
      if (array[l] > array[l+1])
      {
        int temp = array[l];
        array[l] = array[l+1];
        array[l+1] = temp;
      }
```

2. [10 Marks]

Each question has only one valid answer. Write clearly. If we cannot understand your answer, you will not receive any credit (2 marks each):

1. Which code removes the third character from `String s` assuming that `String s` has more than three characters?

   (a) `s = s.substring(3).substring(0, 2);`

   (b) `s = String.valueOf(s, 2);`

   (c) `s = s.substring(0, 2) + s.substring(3);`

   (d) `s = s.substring(0, 1) + s.substring(2);`

2. The *worst-case* time to search in a *sorted* singly-linked list with n items is

   (a) O(1)
   (b) O($\log n$)
   (c) O(n)
   (d) O($n \log n$)

3. The *average* time to search in an *unsorted* singly-linked list with n items is

   (a) O(1)
   (b) O($\log n$)
   (c) O(n)
   (d) O($n \log n$)

4. The *average* time to search in a *sorted* singly-linked list with n items is

   (a) O(1)
   (b) O($\log n$)
   (c) O(n)
   (d) O($n \log n$)

5. Which of the following code creates a non-rectangular two-dimensional *ragged array*?

   (a)
   ```
   int[][] a = new int[10][];
   for (int i = 1; i <= a.length; i ++)
       a[i] = new int[i];
   ```

   (b)
   ```
   int[][] a = new int[1][];
   for (int i = 1; i <= a.length; i ++)
       a[i-1] = new int[i];
   ```

   (c)
   ```
   int[][] a = new int[10][];
   for (int i = 1; i < a.length; i ++)
       a[i] = new int[i+1];
   ```

   (d) All of the above.

3. [15 Marks]

Each of the following three short questions is worth five marks.

(a) Consider the following Java code to implement QuickSort using the `Queue` class taught in the lectures:

```
public static void quickSort (int[] a)
{
  Queue q = new Queue();
  q.enqueue(0);
  q.enqueue(a.length - 1);

  while (!q.isEmpty())
  {
    int left = q.dequeue();
    int right = q.dequeue();

    // Partition it (method not shown)
    int mid = partition(a, left, right);

    if (left < mid - 1)
    {
      q.enqueue(left);
      q.enqueue(mid - 1);
    }
    if (mid + 1 < right)
    {
      q.enqueue(mid + 1);
      q.enqueue(right);
    }
  }
}
```

Will the code still work if we make *all* the following changes? Explain in 1-2 lines to receive full credit.

i) Change `Queue q = new Queue()` to `Stack q = new Stack()`,

ii) change all occurrences of `enqueue()` to `push()`, and

iii) change all occurrences of `dequeue()` to `pop()`.

(b) Given the declaration

```
String s = "Hello There";
```

evaluate each of the following expressions:

(i) `s.charAt(4)`

(ii) `s.indexOf('e')`

(iii) `s.substring(6)`

(iv) `s.equals("Here")`

(v) `s.substring(1,4)`

(c) Do the following base conversions. For full credit, you will need to show your work.

(i) Convert 11111_2 to base 16

(ii) Convert 01220_3 to base 10

(iii) Convert 95_{10} to base 8

(iv) Convert $B1_{16}$ to base 8

(v) Convert 1111_2 to base 3

4. [20 Marks]

(a) (6 marks) Write a *single* line of Java code to delete a node in a linked list. Assume that variable `current` is defined as a `Node` reference and correctly refers (points) to the node before the node to be deleted. Assume that the list is not empty, and the node to be deleted exists.

(b) (7 marks) Implement a *recursive* method `int sumOfDigits(int n)` in Java that returns the sum of all the digits of n. For example, `sumOfDigits(132)` returns `6`. Do not use loops, any helper methods and any Java String methods in your code. Your solution should be less than 10 lines of code, excluding lines containing just a brace bracket, to receive credit.

```
public static int sumOfDigits(int n)
{

}
```

(c) (7 marks) Using string methods, write a *recursive* method that determines if a string is a palindrome (i.e. the string is the same reversed as forward, e.g. "level" or "noon"). You may assume that the string passed to your method contains only lower-case letters. Your solution should be less than 8 lines of code, excluding lines containing just a brace bracket, to receive credit. Do not use any helper methods.

```
public static boolean isPalindrome(String s)
{




}
```

5. [15 Marks]

Assume the linked list `List` (see Java code below) contains integers in ascending order. The list may contain duplicates of the same integer. Write an instance method `deleteAll(int x)` to delete all occurrences of the value `x` from the list. Your solution *must* use the following strategy:

- Identify the first node (if any) that contains `x` to remove (call it `nodeA`), then
- find the first node down in the list from `nodeA` that does not contain `x` (call it `nodeB`), and
- adjust the reference that points to `nodeA` so that it points to `nodeB` instead.

Do not assume existence of other methods, so you must write the code for all methods you need. Your solution should be less than 18 lines of code, excluding lines containing just a brace bracket, to receive any credit.

```
class List
{
   private class Node
   {
      public int data ;
      public Node next ;
   }
   Node head;

   public void deleteAll(int x)
   {

   }
}
```

6. [10 Marks]

Ackerman's function is defined recursively on non-negative integers m and n as follows:

```
a(m, n) = n+1                    if m == 0
a(m, n) = a(m-1, 1)              if m != 0, n == 0
a(m, n) = a(m-1, a(m, n-1))      if m != 0, n != 0
```

(a) (3 marks) Calculate the value of `a(2, 2)`. Show your work.

(b) (7 marks) Write a single *recursive* method in Java that computes Ackerman's function. Do not use other methods or loops or you will receive no credit. Assume parameters m and n are always greater or equal to zero. Your solution should be less than 12 lines of code, excluding lines containing just a brace bracket, to receive credit.

```
public static int myAckerman(int m, int n)
{

}
```

7. [10 Marks]

The algorithm outlined below sorts a set of n integers in the range `[1...k]`. The basic idea of this algorithm is to determine for each value x the number of elements less than x. This information can be used to place x directly into its correct position in the output array. For example, if there are 7 elements less than x, then x belongs in output position 8.

In the Java code that follows, we assume that the input array with n integers is `A[1...n]`. In other words, `length(A)=n+1` and the first entry `A[0]` is not used. We also assume that the output array which will contain the sorted sequence is `B[1...n]`. Array of integers `C[1...k]` provides temporary storage during the operation of the algorithm. The numbers on the left of the statements indicate line program numbers and they are not part of the code.

```
   public static void countingSort(int[] A, int[] B, int k)
   {
0.      int[] C = new int[k+1];
1.      for(int i = 1; i <= k; i++)
2.         C[i]=0;
3.      for(int j = 1; j <= A.length-1; j++)
4.         C[A[j]] = C[A[j]] + 1;
5.      /*  C[i] now contains the number of elements equal to i  */
6.      for(int i = 2; i <= k; i++)
7.         C[i] = C[i] + C[i-1];
8.      /*  C[i] now contains the number of elements less than or equal to i */
9.      for(int j = A.length-1; j >= 1; j--)
10.     {
11.        B[C[A[j]]] = A[j];
12.        C[A[j]] = C[A[j]] - 1;
13.     }
    }
```

Assume input array `A[1...8]` is as shown below, containing integers in the range `[1...6]`. Fill all *applicable* values in the boxes during the execution of the algorithm **(a)** after line 4; **(b)** after line 7; **(c)/(d)/(e)** after one/two/three iteration(s) of the `for` loop in lines 9–13.

8. [20 Marks]

In this question, you will implement a linked-list class to maintain a list of students.

(a) (10 marks) First, create a class named `Student`. Each `Student` object will store first name, last name, and a student number represented by an `long` type integer. Provide accessor and mutator methods for each and a constructor that takes these three values as parameters.

Additionally, write a `compareTo()` instance method that compares the students by lastName, then firstName and then by student number (i.e., two students with the same last name would be compared according to their first names, and two students with identical first and last names would be compared according to their student number). The method should return `-1` if the implicit student comes first, `0` if the two students compare equal and `1` otherwise.

Finally, provide a `toString()` instance method that returns a string containing the `Student`'s information in the format "lastname, firstname - studentNumber".

Continue your solutions to Question 8(a) on this page.

(b) (10 marks) Create a class named `StudentList` that stores `Student` objects in a singly-linked list. You are to let Java provide the default constructor. You should define the following methods for the `StudentList` class:

(i) (3 marks) An instance method `addStudent` that inserts a new `Student` object in the correct location to keep the list sorted in ascending order, according to the `compareTo` method previously defined in the `Student` class.

(ii) (2 marks) An instance method `printList` that prints the elements in the list, one element per line, using the `toString` method previously defined in the `Student` class.

(iii) (2 marks) An instance method `isEmpty` that returns `true` is the list is empty, and `false` otherwise.

(iv) (3 marks) An instance method `findByStudentNumber`, that takes a `long` type student number as parameter, and returns a `String` that represents the information of the student with the given student number. The `String` to be returned is formatted using the `toString` method previously defined in the `Student` class. If no `Student` object in the list has the desired number, then the method should return `null`.

Continue your solutions to Question 8(b) on this page.

Extra space *Indicate clearly which question(s) you are answering on this page.*

Extra space *Indicate clearly which question(s) you are answering on this page.*

Extra space *Indicate clearly which question(s) you are answering on this page.*