Department of Computer Science
Graduate Exam in Programming
Spring 2016

You must write Java or C++ on the first page of your answers!! You may answer the questions using either Java or C++, but you must use the same language for each question on the exam.

1. (2 pts) Write a function named checkPassword that takes a string parameter and returns true if it meets the following criteria:
   • it is at least 8 characters long
   • it contains at least one of these three characters: ‘!’ ‘#’ ‘$’
   If it does not meet both of these criteria, it should return false.

2. (1.5 pts) Write a recursive function named sumOver that takes an int parameter n and returns a double value which is the sum of the reciprocals of the first n positive integers. The reciprocal of x is 1.0/x. For example, sumOver(3) returns 1.833 which is 1.0/1 + 1.0/2 + 1.0/3. Your function must work for values of n greater than or equal to 0. By definition, sumOver(0) = 0.

3. (2.5 pts) Given a linked list class defined as follows:

   ```
   // C++
   class List {
   private:
     struct Node {
       double value;
       Node *next;
     };
     // points to the first node
     Node *head;
   public:
     List() {head=NULL;}
     void insertB4Last(double x);
   }
   // Java
   class List {
   private class Node {
     double value;
     Node next;
   }
     // refers to the first node
     private Node head;
   public List() {head = null;}
   public void insertB4Last(double x) {
   };
   ```

Write the member function insertB4Last(double x) which adds a Node containing x to the list so that it immediately precedes the last node. If the list is empty or contains only one Node, your function should do nothing (do not change the list). Only perform the insertion if the list has at least two Nodes.
4. (4 pts) A Hash Table stores a finite set of values of a given type in an array. It uses a hash function that takes a value of the given type as an argument and computes from it a position in the array. The problem is that multiple values may have the same hash value.

One solution to this problem is linear probing. When trying to insert a value $x$, if another value is already in the position computed by the hash function (say position $p$), then place $x$ at position $p+1$, unless that location is already full, in which case try $p+2$. Continue until an empty location is found (and if the end of the array is reached, go back to the beginning of the array, position 0).

When trying to determine if a value $x$ is already in the Hash Table, you must also use linear probing if a value not equal to $x$ is found in the position computed by the hash function.

The following simple HashTable class stores a set of non-negative integers. The hash function is given, and the size can be any value less than or equal to 10000:

```cpp
// C++
class HashTable {
private:
    int array[10000];
    int size;
    int hash (int key) {
        return key % size;
    }
public:
    HashTable (int s);
    void insert (int x);
    bool find(int x);
};
```

```java
// Java
class HashTable {
    private int array[];
    private int size;
    private int hash (int key) {
        return key % size;
    }
    public HashTable (int s) {...}
    public void insert (int x) {...}
    public boolean find(int x) {...}
};
```

Implement the three public functions:
• a 1-argument constructor that creates an empty HashTable, given a size. All array values should be set to -1 to indicate that they are empty.
• `insert(x)` to put integer $x$ in the table. Use the hash function and linear probing to find its proper position. Do not add an element that is already in the HashTable.
• `find(x)` to return `true` if $x$ has been inserted into the table, `false` otherwise. Use the hash function and linear probing to determine the result.