Computer Engineering Department
PhD Qualifier Exam

31 January, 2014

- Time: 10:00 - 13:00
- The exam contains 8 questions in total. You are to choose 6 of these questions and answer only to them.
- By submitting this exam, you agree to fully comply with Koç University Student Code of Conduct, and accept any punishment in case of failure to comply. If you do not submit this exam on time, you will directly fail the qualifier.
- The exam is open-book and open-notes.
- You are not allowed to use any electronic equipment such as computers and mobile phones.
- In general, you are not allowed to ask questions during the exam.
- Good luck!

1. **a)** Write a recursive method to reverse a singly-linked list (in a Java like language). Your method should reverse the list without allocating new nodes, just using the nodes in the list.

```
// example
// let l be a list (0,1,2,3)
l =  recReverse(l)
// l is now (3,2,1,0)

class Node {
   int data
   Node next;
}


//  head is the reference to the first node in the list
//  recReverse returns the reference to the first node in the reversed list
Node recReverse(Node head) {
    // your code goes here









}
```

b) Given an array of n numbers, A, we want to find if there are two numbers $A[i]$ and $A[j]$ in the array such that $A[i] + A[j] = k$ for a given k, and $i \neq j$. For instance, if the input array is $[4, 7, 2, 3, 5]$ and k is 10, the answer is TRUE, that is, there are two elements such that their sum is k ($A[1] + A[3] = 10$). Suggest a **subquadratic** algorithm to solve the problem. Describe your algorithm in pseudo-code. You can use any data structure or algorithm, you don't need to explain them. For example, if you use a stack, just declare a stack and use stack methods/operations.

i) Pseudo-code here:

ii) What is the running time of your algorithm?

2. A coin factory produces an assortment of coins: 50% of them are normal coins, 10% are double headed coins, and 40% are double tailed coins. You see one coin produced by this factory on the street and its top face is showing tails. What is the probability that it is a normal coin?

3. Let R be the relation on the set of functions from Z^+ to Z^+ such that (f, g) belongs to R if and only if f is $O(g)$. By using the definition of big-O notation, prove or disprove that R is an equivalence relation. Note that Z^+ stands for the set of positive integers.

4. **a) Scheduling.**
Consider **multi-level feedback queue scheduling** in a single-processor system: Queue-1, the first level queue, is given a quantum of 4 ms, Queue-2 is given a quantum of 8 ms, Queue-3 is given a quantum of 12 ms, and Queue-4 is scheduled FCFS. Assume all processes arrive in the following order (at time zero) to Queue-1 with the CPU burst times (in ms). Queue-1 has a feedback to Queue-2; Queue-2 has a feedback to Queue-3; and Queue-3 has a feedback to Queue-4.

- Give the **Gantt chart** for the described scheduling, and calculate the **average waiting time**. Recall that a Gantt chart illustrates processes' schedule in time, and waiting time of a process is defined as the amount of time the process has been waiting in the ready queue.
- Show the **preemptions** on your Gantt chart in part, and give the number of preemptions for the schedule.

Process	Burst time
P1	10
P2	20
P3	27
P4	18
P5	30

b) Process synchronization.

Servers can be designed to limit the number of open connections. Consider a server that needs to have at most K socket connections at any point in time. As soon as K connections are made, the server will not accept another incoming connection until an existing connection is released.

As a solution to this synchronization problem, give an algorithm (for a server process) using **semaphore** primitives to limit the number of concurrent connections. Explain how your algorithm solves the problem.

5. Provide an algorithm that determines whether a given directed graph G contains a cycle or not. Argue that your algorithm is correct, i.e., your algorithm declares that G has a cycle, if and only if it really does have a cycle. Argue that your algorithm terminates and analyze the computational complexity of your algorithm.

6. Using an object-oriented language you are familiar with, write code for a class named `Account`. This class should represent a simple bank account for a person and should have fields that store the account holder's name, account balance, and *a unique account number*. Accounts should be numbered starting from 1 and should be assigned account numbers automatically so that, for instance, the 11th account object created has account number 11. Make all fields of `Account` private. All accesses to data stored in an `Account` object should be using methods.

   Write code for the following:

   - Implement a method `getNumberOfAccounts` that returns the number of `Account` objects that exist.
   - Implement a method `transferFrom(Account otherAccount, double transferAmount)` that transfers `transferAmount` amount of money from `otherAccount` to the `Account` object on which the method is invoked. The method should throw an `InsufficientFundsException` if `otherAccount`'s balance is less than `transferAmount`.

7. There are four major types of parameter passing conventions: call by value, call by reference, call by need, and call by name. Write a procedure that prints out which one of these parameter passing mechanisms is in use in a given language system. For each case, explain your reasoning and how your program works in detail. You can use the EREF or IREF syntax. You can also use a conventional language of your choice, but in that case clearly explain the semantics of your expressions.

8. **a)** Consider the following database schema which describes flights between cities, and tickets issued by travel agents:

   flight(<u>fno</u>, from, to, departureTime, arrivalTime)
   ticket(<u>tid</u>, travelAgent, passenger)
   itenary(<u>tid</u>,fno)

   Use both relational algebra and SQL to answer the following query: *Find all possible trips from Istanbul to SanFransisco, which consists of two connecting flights. Flights connect if flight 1 arrives at the airport from where flight 2 leaves and the arrival time of flight 1 is less than the departure time of flight 2.* A trip is composed of two flights: flight no 1, flight no 2.

   RA solution:

   SQL solution:

b) Consider a relation $R(A, B, C)$.

i) Write a single relational algebra expression that returns an empty result *if and only if* the functional dependency $A \rightarrow B$ holds on R.

ii) Write one SQL SELECT statement for the same problem.

Assume that we have the following four tuples in the relation $R(A, B, C)$: $(1, 2, 3), (4, 2, 3), (5, 3, 3), (5, 3, 4)$. Which of the following you can infer *does not hold* over R? Give short one-line reasons, for each.

iii) $A \rightarrow B$

iv) $BC \rightarrow A$