Algorithms and Data Structures (60-254)
First Midterm Exam, Summer 2018, June 08
Total Marks : 40
Time: 120 mins

Instructions:

- This is a **no-calculator, closed-notes, closed-book** exam.
- Unless otherwise stated, all logs are to the base 2
- Answer **all** the questions.

Name: Tyler Hedges-Johnston **SID:** 105213585

Qn#	Part 1	Part 2	Part 3
Max Marks	10	10	20
Marks Obtained			

Part I: Multiple choice (15 marks = 1 marks × 5 + 2 marks × 5)

For this part *clearly* write down the single choice that you believe is the most accuarate.

1. Let $m = 34$ and $n = 21$ for the algorithm GreatestCommonDivisor below:

```
Algorithm 1 GreatestCommonDivisor
Input: Positive integers $m$ and $n$
Output: Greatest common divisor of $m$ and $n$
1: $r \leftarrow m \bmod n$    34 mod 21
2: while $r \neq 0$ do    13
3:     $m \leftarrow n$
4:     $n \leftarrow r$
5:     $r \leftarrow m \bmod n$
6: end while
7: return $n$
```

The number of times m and n are updated inside the while loop is:

[✓ (a) 6]
(b) 7
(c) 8
(d) None of the above.

Ans: a

2. If $T(n) = 5 * n^3 - 6 * n + 12$ is an estimate of the time-complexity of some algorithm for an input of size n, which of the following is a correct estimate of $T(n)$ in terms of the big-Oh notation:

(a) $T(n) = O(n)$
(b) $T(n) = O(n^2)$
[✓ (c) $T(n) = O(n^3)$]
(d) None of the above.

Ans: c

3. Let $T(n)$ be the time-complexity of the divide-and-conquer algorithm (as described in class) for finding a maximum contiguous subsequence of a sequence of n integers. Then it can be shown that $T(n)$ is best described by:

   (a) $T(n) = O(n)$
   [✓ (b)] $T(n) = O(n \log n)$
   (c) $T(n) = O(\log n)$
   (d) None of the above.

   Ans: b

4. The second algorithm we discussed in class for finding a maximum contiguous subsequence of n integers solved the problem in $O(n)$ time (or, linear time as we say) by resetting the current candidate start index i for a maximum contiguous sequence to a new candidate start index i whenever a negative prefix is found.

   1 2 3 4 5 6 7 8 9
   Given the list of integers 1, 2, 3, -7, 8, 9, -18 , 10, 11, indexed from 1 to 9, during the left-to-right scan, this algorithm will set as candidate start positions the elements at the index positions:

   (a) 8
   (b) 1, 8
   [✓ (c)] 1, 5, 8
   (d) None of the above

   Ans: c

5. In lab 2, you implemented an algorithm to check if a parentheses sequence made up of opening parenthesis, '(', and closing parenthesis, ')', is balanced or not. Thus, if the input parentheses sequence to your algorithm is "(())" it would return "true" and "false" if the input were, for example, "))((".

   Now suppose the input parentheses sequence were: "()(())()"; how many times during the left-to-right scan of this sequence will the stack become empty before your algorithm returns the answer: "true"?

(a) 1 time.
(b) 2 times.
(c) 3 times.
[✓ (d) None of the above.]

Ans: d

6. The Fibonacci sequence 1,1,2,3,... can be generated iteratively by starting with the first two (0th and 1st) and generating every subsequent one by adding the previous two. In fact, the nth Fibonacci number can be generated by $n-1$ additions for $n \geq 2$. Thus we need 5 additions to generate the $6th$ Fibonacci number. However, a recursive program to generate the $8th$ Fibonacci number needs:

   (a) 31 additions
   (b) 32 additions
   (c) 33 additions
   [✓ (d) None of the above]

   Hint: You can make use of the observation that if A_n be the number of additions needed to compute the n-th Fibonacci number, then it satisfies the following recurrence relation:

$$\begin{aligned} A_n &= A_{n-1} + A_{n-2} + 1 \text{ for } n \geq 2 \\ &= 0 \text{ for } n = 0 \text{ and } n = 1 \end{aligned}$$

   Ans: d

7. Consider the following recursive C-function, `mystery(n)`, where the integer argument n is a non-negative integer.

```
int mystery(int n){
        if (n == 0) return 1; // base case
        else if (n == 1) return 0; // base case
        else if (n % 2 == 0) return 1 + mystery(n/2);
        else return mystery(n/2);
}
```

When called with the value $n = 25$ from `main` this function will return a value of:

(a) 1
[✓ (b) 2]
(c) 3
(d) None of the above

Ans: b

8. For lab 2 you implemented an algorithm to compute a maximum contiguous subsequence (*mcs*) of a sequence of integers, $list[1:n]$, by computing the *start* of an *mcs* that ends at each of the elements of $list[1:n]$. This is done in a left-to right scan, recording for each element the *start* position and the sum of the *mcs* that ends on this element.

The algorithm is based on the observation that:

$$sum[i] = \text{maximum}\{sum[i-1] + list[i], list[i], 0\},$$

where $sum[i]$ is the sum of the *mcs* that ends at the i-th element, $list[i]$, of $list[1:n]$, and the *start* index of the i-th element is reset (to what ?) from the initial value of -1 if the *maximum* is one of the first two arguments. Assume that $sum[0] = 0$ by definition.

The above algorithm was applied to the $list[1..5] = \{-83, 26, -37, -63, 85\}$, and the following table was generated.

index	1	2	3	4	5
list	-83	26	-37	-63	85
start index	-1	2	a	-1	c
mcs sum	0	26	b	0	d

The missing entries a, b, c, d are:

(a) 1, -11, 2, 111, respectively
(b) 2, -11, -1, 0, respectively
[✓ (c) -1, 0, 5, 85 respectively]

(d) None of the above.

Ans: c

9. Consider the following recursive function, `power(base, expo)`, written in C.

```
int power(int base, int expo){
        int count;

        if (expo == 1) {count = 0; return count;}
        else if (expo % 2 == 1) {count = 1 + power(base, expo-1); return count;}
        else {count = 1 + power(base, expo/2); return count;}
}// power
```

What is the value this function will return when it is called from the `main` function with the parameter values `base = 2` and `expo = 25` ?

(a) 5
[✓ (b) 6]
(c) 7
(d) 8

Ans: b

10. In class, we discussed a divide-and-conquer approach to determine a maximum contiguous sub-sequence of a sequence of integers, S. The main steps of this algorithm are as follows:

[✓ (a) Divide the sequence S into two nearly equal halves, S_l and S_r.]
(b) Find a maximum contiguous sub-sequence of each of S_l and S_r, using the divide-and-conquer principle recursively.
(c) Find a maximum contiguous sub-sequence that straddles S_l and S_r. Such a straddling sequence is made up of a suffix of S_l and a prefix of S_r.
(d) Return the maximum of the three maximum contiguous sub-sequences

The crucial part of the algorithm is how the straddling sequence is found (third step). The sequence below:

$$-1, 10, -3, 8, -5, 6, -7, 4, -9, 2$$

can be divided into a left-half -1, 10, -3, 8, -5 and a right-half 6, -7, 4, -9. 2.

Pick the correct answer. The maximum contiguous subsequence of the full sequence is:

[✓ (a)] contained in the left-half
(b) contained in the right-half
(c) straddles across the left and right-halves.

Ans: A

Part II: True/False answers (10 marks = 2 marks × 5)

For this part, *clearly* write down if each asserted statement is TRUE or FALSE. You must write "TRUE" or "FALSE" not "T" or "F".

1. Let $r_0, r_1, r_2, ..., r_k$ be the sequence of remainders, when you apply Euclid's GreatestCommonDivisor algorithm to a pair of positive integer inputs m and n.

   Assertion: The algorithm terminates because the r_i's are *strictly decreasing* and that $r_i \geq 0$ for each i.

   Ans: True

2. Let $f(n) = n^2+n+1$ and $g(n) = n^2 - 1$ be integer-valued functions of a non-negative integer n.

   Assertion $f(n) = \Theta(g(n))$.

   Ans: True

3. Let $a_1, a_2, ..., a_n$ be a sequence of integers. A *suffix* of this sequence is a subsequence $a_i, a_{i+1}, .., a_n$, where $i \leq n$. For example, if we have a sequence 3, 4, 5, -2, 7, -8, 9, then 7, -8, 9 is a suffix; -8, 9 is another and so on.

   This question refers to the maximum sum contiguous subsequence problem.

   Assertion: A contiguous subsequence that has maximum sum cannot have a negative suffix.

   Ans: False

4. Recall the Tower of Hanoi problem. Let $M(n)$ be the minimum number of single disk-moves that are sufficient to transfer n disks from peg A to peg C, following all the rules for moving a disk from one peg to another.

   Assertion: When $n = 5$, $M(5) = 31$

   Ans: True

5. This question relates to Lab 1. Let $r_0, r_1, r_2, ..., r_k$ be the sequence of remainders, when you apply Euclid's GCD algorithm to the inputs m and n. Each r_i can be written as a linear combination of m and n. Using this idea, we can compute integers u and v such that $um + vn = \gcd(m, n)$

Assertion: The integers u and v are both positive.

Ans: True