

COMPSCI 320SC 2017 Midterm Test

Attempt *all* questions. (Use of calculators is NOT permitted.)

Put the answers in the space below the questions. Write clearly and *show all your work!*

Marks for each question are shown below and just before each answer area.

This 50 minute test is worth 10% of your final grade for the course.

| | | | | | |
|------------------------|---|---|---|---|-------|
| Question #: | 1 | 2 | 3 | 4 | Total |
| <i>Possible marks:</i> | 6 | 6 | 6 | 6 | 24 |
| <i>Awarded marks:</i> | | | | | |

University ID: _____

Student Name: _____

Student Signature: _____

Time Finished: _____

1. For each of the following statements. State whether it is True/False (1 mark) and justify your answer (2 marks).

(a) $\frac{\lg n}{\sqrt{n}}$ is not $O(\sqrt{n})$. **(3 marks)**

(b) If $f(n) \in O(n)$ then $f(n)^2 \in O(n^2)$. **(3 marks)**

2. Consider the following Python “divide-and-conquer” function:

```
def excited(L,n):  
    if n <= 2: return  
    if n == 3: return excited('*'+L+'*',4)  
    print(L)  
    return excited(L, n//2)
```

Let $T(n)$ denote the number of lines of output generated by a call of `excited(L,n)`.

(a) Provide a recurrence equation for $T(n)$. (2 marks)

(b) What are the values of $T(3)$, $T(4)$, $T(5)$, $T(6)$, $T(7)$ and $T(8)$? (2 marks)

(c) Solve the recurrence exactly for n being a power of 2 (i.e., $n = 2^k$ for $k > 1$). (2 marks)

3. (a) State the Red Rule (hint: cycle) and Blue Rule (hint: cutset) and the generic algorithm we used as a basis for proving correctness of several greedy Minimum Spanning Tree (MST) algorithms. **(3 marks)**

- (b) Briefly explain Prim, Kruskal and Boruvka's MST algorithms in terms of the Red and Blue Rules. **(3 marks)**

4. Below are two Python functions that try to evaluate a polynomial $p(x) = a_0 + a_1x + a_2x^2 + \dots + a_nx^n$ of degree n . For each case indicate the exact number of multiplications and additions in terms of n (2 marks) and state True/False whether each correctly evaluates the polynomial at x (1 mark).

(a)

```
def eval1(a,n,x):  
    p,xpwr = a[0],x  
    for i in range(1,n):  
        p += a[i]*xpwr  
        xpwr *= x  
    return p + a[n]*xpwr
```

(3 marks)

(b)

```
def eval2(a,n,x):  
    p = a[n]  
    for i in range(n-1,0,-1):  
        p = p*x + a[i]  
    return p*x + a[0]
```

(3 marks)

