COMPSCI 320SC 2018 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	Total
Possible marks:	10	10	10	30
Awarded marks:				

University ID: ______________________________

Student Name: ______________________________

Student Signature: ______________________________

Time Finished: ______________________________

1. Consider a stable-matching problem with the following preference lists:

R	**1**	**2**	**3**	**4**
Ahi	Wini	Xuan	Yasi	Zina
Bob	Xuan	Wini	Yasi	Zina
Cy	Wini	Xuan	Yasi	Zina
Dax	Wini	Xuan	Yasi	Zina

S	**1**	**2**	**3**	**4**
Wini	Cy	Ahi	Bob	Dax
Xuan	Dax	Bob	Ahi	Cy
Yasi	Bob	Dax	Cy	Ahi
Zina	Ahi	Cy	Dax	Bob

Recall that a matching is unstable if there exists any unmatched pair who prefer each other to their current partners. Also recall that the Gale-Shapley algorithm iterates over all currently-unengaged $r \in R$, with each r proposing to their highest-preference $s \in S$ until they find an s who is either unengaged, or who prefers them to their current fiancé f.

(a) What matching is produced by the Gale-Shapley algorithm for these preferences? Show your working: you will get 0 marks if you exhibit a matching without briefly explaining how you found it. ***(4 marks)***

Proposals are made in the following order (queue-based): $A- > W, B- > X, C- > W, D- > W, A- > X, A- > Y. D- > X, B- > W, B- > Y, A- > Z$ so final matchings, $A = Z, B = Y, C = W, D = X$.
Or proposals are made in the following order (stack based): $A- > W, B- > X. C- > W, A- > X, A- > Y, D- > W, D- > X, B- > W, B- > Y, A- > Z$ with same final matchings as above.

(b) Consider the following method `prefer(s,r,f)` of determining whether a given $s \in S$ prefers a potential match $r \in R$ to their current fiancé $f \in R$: scan s's preference list, returning `True` if r appears earlier in the list than f. What is the worst-case (big-Oh) runtime of the Gale-Shapley algorithm if this implementation of `prefer()` is used, and if there are $n = |R| = |S|$ elements in each set? Explain your reasoning briefly: you will get no marks for writing down a big-Oh expression with no explanation. ***(3 marks)***

The `prefer()` method takes $O(n)$ time, because it is scanning sequentially through a list of length n. The worst case occurs when both r and f are near the end of s's list. The Gale-Shapley algorithm may examine most of the n^2 elements in the preference lists for R if there are a lot of broken engagements, so the runtime is $O(n^3)$. Award one mark for the analysis of `prefer()`, one mark for the iteration count of Gale-Shapley, and one mark for multiplying these two big-Oh values.

(c) The Gale-Shapley algorithm is unsuitable for use in some applications, because it is biased toward its first set R. Is this bias apparent for the given input? Explain briefly. To receive full marks, you must show a second stable matching for this input which is clearly more favourable to elements in S. ***(3 marks)***

Here is another copy of the input:

R	**1**	**2**	**3**	**4**
Ahi	Wini	Xuan	Yasi	Zina
Bob	Xuan	Wini	Yasi	Zina
Cy	Wini	Xuan	Yasi	Zina
Dax	Wini	Xuan	Yasi	Zina

S	**1**	**2**	**3**	**4**
Wini	Cy	Ahi	Bob	Dax
Xuan	Dax	Bob	Ahi	Cy
Yasi	Bob	Dax	Cy	Ahi
Zina	Ahi	Cy	Dax	Bob

No apparent bias since all women get their first choice.

2. Consider the following divide-and-conquer "algorithm":

function foo(int n)
  for $i = 1$ **to** n **do**
    $j = 1$
    while $j * j < n$ **do**
      printline "bar"
      $j = j + 1$
  if $n > 0$ **then**
    for $k = 1$ **to** 7 **do**
      foo($\lfloor n/4 \rfloor$)

Let $T(n)$ denote the number of lines of output generated by a call of foo(n).

(a) What are the values of $T(4)$, $T(12)$ and $T(16)$? ***(3 marks)***

T(1)=0, T(2)=2, T(3)=3,
T(4)=4*1+7*T(1)=4,
T(12)=12*3+7*T(3)=36+21=57, and
T(16)=16*3+7*T(4)=48+28=76

(b) Provide a recurrence equation for $T(n)$. ***(4 marks)***

$$T(n) = \begin{cases} 0 & \text{if } n = 0 \\ 7 \cdot T(n/4) + \Theta(n\sqrt{n}) & \text{if } n > 0 \end{cases}$$

(c) Solve the recurrence asymptotically for general n.

You may want to make use of the following 'master recurrence theorem':
Assume $T(n) = a \cdot T(n/b) + \Theta(n^c)$ is the total time for a divide-and-conquer algorithm then:

$$T(n) \in \begin{cases} \Theta(n^c) & \text{if } a < b^c \\ \Theta(n^c \log n) & \text{if } a = b^c \\ \Theta(n^{\log_b a}) & \text{if } a > b^c \end{cases}$$

(3 marks)

With $a = 7$, $b = 4$ and $c = 1.5$, so with $a = 7 < 8 = b^c$, we get $T(n) = \Theta(n^{1.5})$ from the master recurrence theorem.

3. (a) Explain Karatsuba's integer multiplication algorithm and indicate why it runs in time $O(n^{\lg 3}) = O(n^{1.585})$, where n is the number of bits of each of the two input integers.
Hint: $bc + ad = ac + bd - (a-b)(c-d)$. **(*5 marks*)**

For $x = x_1 \cdot 2^{n/2} + x_0$ and $y = y_1 \cdot 2^{n/2} + y_0$, the Karatsuba algorithm recursively computes $p = M(x_1 + x_0, y_1 + y_0)$, $x_0y_0 = M(x_0, y_0)$, and $x_1y_1 = M(x_1, y_1)$ and returns $x_1y_1 \cdot 2^n + (p - x_1y_1 - x_0y_0) \cdot 2^{n/2} + x_0y_0$.
Then use the divide and conquer master theorem for the recurrence $T(n) = 3T(n/2) + O(n)$.
(Two marks for showing how to split x and y in half, One mark for showing return value. Two marks for applying D&C master theorem.)

(b) We are given a collection C of $n > 1$ car parts, and also a mechanic expert $\mathcal{E}$ddy that can decide whether two parts belong to the same brand of car (e.g. Ford, Holden, Nissan, Toyota, ...). Describe a divide-and-conquer algorithm that decides if at least $\lfloor n/2 \rfloor + 1$ of the parts in C are the same brand by asking at most $O(n \lg n)$ questions to $\mathcal{E}$ddy. You can only make an inquiry to the mechanic with two parts at a time and $\mathcal{E}$ddy just says 'yes/no' and does not indicate any brand names. **(*5 marks*)**

Divide group of parts into roughly two halves and recursively obtain up to two potential majority brands (2 marks). Then, with each representative, use $O(n)$ questions to the expert to decide if any of these are actual majorities (2 marks). From the recurrence $T(n) = 2T(n/2) + O(n)$ we get the desired result (1 mark).