National University of Singapore

School of Computing

CS3230 - Design and Analysis of Algorithms

Midterm Assessment

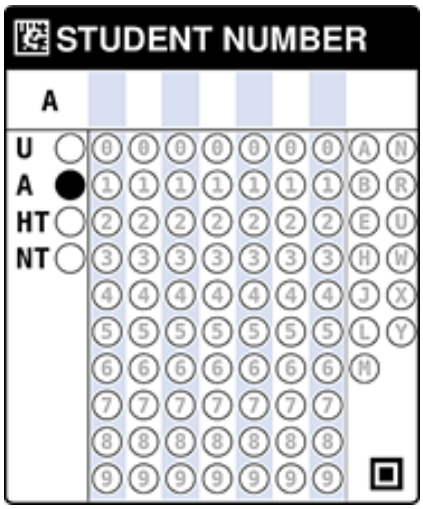
(Semester 2 AY2023/24)

Time Allowed: 80 minutes

INSTRUCTIONS TO CANDIDATES:

1. Do **NOT** open this assessment paper until you are told to do so.
2. This assessment paper contains TWO (2) sections. It comprises EIGHT (8) printed pages.
3. This is an **Open Book** Assessment.
4. For Section A, use the bubbles on page 4 (use 2B pencil).
5. For Section B, answer **ALL** questions within the **boxed space**.
   If you leave the box blank, you will get an automatic 0.5 mark (**NOT** for Bonus question).
   However, if you write at least a single character and it is totally wrong, you will get a 0 mark.
   You can use either a pen or a pencil. Just make sure that you write **legibly**!
6. Important tips: Pace yourself! Do **not** spend too much time on one (hard) question.
   Read all the questions first! Some questions might be easier than they appear.
7. You can assume that all **logarithms are in base** 2.

Write your Student Number in the box below using **(2B) pencil**:



A Multiple Choice Questions ($10 \times 1.5 = 15$ marks)

Select the **best unique** answer for each question. Each correct answer is worth 1.5 marks.
Write your MCQ answers in the special MCQ answer box on page 4 for automatic grading. We do not manually check your answer.

The default answers ~~(as the MCQ section is not supposed to be archived to open up possibilities of reuse in the future)~~: ccba dcce cc.

1. Which of the following statements is **TRUE**?

   a). $n! = \Omega(n^n)$

   b). $2^n = O((\log n)^{\log n})$

   c). $n/\log n = \Omega(n^{0.999})$ Answer.

   d). $n^{\log n} = O(n^{2000})$

   e). None of the above

2. $(2n)!$ is in

   a). $O((n!) \cdot (n!))$

   b). $O((n!) \cdot (2^n))$

   c). $O((n!) \cdot (n!) \cdot (n!))$ Answer.

   d). $\Omega((n!) \cdot (n!) \cdot (n!))$

   e). None of the above

3. While computing the time complexity of a recursive algorithm, you get the following recurrence relation: $T(n) = 49T(n/7) + n^{7/2}$. Then which one of the following is **TRUE**?

   a). $T(n) = \Theta(n^{7/2} \log n)$

   b). $T(n) = \Theta(n^{7/2})$ Answer. Case 3 of master theorem, regular, thus $T(n) = \Theta(n^{7/2})$.

   c). $T(n) = o(n^2)$

   d). $T(n) = \Theta(n^2)$

   e). $T(n) = \Theta(n^2 \log n)$

4. Suppose $T(1) = 1$, $T(n) = T(n-1) + 3n + 7$. Then which one of the following is **TRUE**?

   a). $T(n)$ is in $\Theta(n^2)$ Answer.

   b). $T(n)$ is in $\Theta(n^3)$

   c). $T(n)$ is in $\Theta(n)$

   d). $T(n)$ is in $\Omega(2^n)$

   e). None of the above

5. Which of the following statements is **TRUE**?

   a). Longest Common Subsequence of any two arbitrary input sequences is always unique

   b). Any algorithm to compute n-th Fibonacci number (mod n) must take $\Omega(n)$ time

   c). The running time of a dynamic programming algorithm is always upper bounded by the number of distinct subproblems

   d). Any comparison-based algorithm to search an integer in an n-length sorted integer array must take $\Omega(\log n)$ comparisons Answer.

   e). None of the above

6. Consider the recurrence relation: $T(n) = T(0.9n) + n$. (Assume, $T(n) \leq 99$, for $n \leq 17$.) What is the height of the corresponding recursion tree?

   a). $\Theta(1)$

   b). $\Theta(\log \log n)$

   c). $\Theta(\log n)$ Answer. If height is h, then $(0.9)^h n = \Theta(1) \Rightarrow (10/9)^h = \Theta(n) \Rightarrow h = \Theta(\log n)$.

   d). $\Theta(n^{0.1})$

   e). $\Theta(n)$

7. Consider the deterministic algorithm for quick sort. Which of the following statements is **TRUE**?

   a). For all large enough n, for every possible input of n numbers, quick sort algorithm performs at least $0.001n^2$ comparisons

   b). For all constant $c > 0$, for all large enough n, for all possible input of n numbers, quick sort algorithm performs at least cn^2 comparisons

   c). For some constant $c > 0$, for all large enough n, for all possible input of n numbers, quick sort algorithm performs at most cn^2 comparisons Answer.

   d). For some constant $c > 0$, for all large enough n, for all possible input of$n$ numbers, quick sort algorithm performs at most $cn \log n$ comparisons

   e). None of the above

8. Suppose you are given as input an (undirected) graph $G = (V, E)$ with n nodes and m edges, where each vertex has a unique label from $\{1, 2, \cdots, n\}$. Next, consider the following randomized procedure: Select a vertex $v \in V$ uniformly at random and then return the label of v as output. Let X be the random variable denoting the output of this randomized procedure. What is the expected value of X?

   a). $m/2$

   b). $2m/n$

   c). $n/2$

d). $1/n$

e). $(n+1)/2$ Answer. $\mathbb{E}[X] = \sum_{v \in V} label(v) \cdot \Pr[v \text{ is picked}] = \frac{1}{n}(1+2+\cdots+n) = (n+1)/2$.

9. For $n > 0$, Strassen's multiplication algorithm for multiplying two $n \times n$ integer matrices does

a). $O(n^{\log_2 7})$ number of integer multiplications, $\Omega(n^3)$ number of integer additions

b). $\Omega(n^3)$ number of integer multiplications, $O(n^{\log_2 7})$ number of integer additions

c). $O(n^{\log_2 7})$ number of integer multiplications, $O(n^{\log_2 7})$ number of integer additions Answer.

d). $\Theta(n^3)$ number of integer multiplications, $\Theta(n^{\log_2 7})$ number of integer additions

e). None of the above

10. Suppose $Pr(X)$ denotes the probability of event X happening. Suppose A and B are two events. Which of the following statements is always **TRUE**:

a). $Pr(A \text{ or } B) > max(Pr(A), Pr(B))$

b). $Pr(A \text{ or } B) \geq Pr(A) + Pr(B)$

c). $Pr(A \text{ or } B) \leq Pr(A) + Pr(B)$ Answer.

d). $Pr(A \text{ and } B) = Pr(A) \cdot Pr(B)$

e). None of the above

Write your MCQ answers in the answer box below using (2B) pencil:

No	1	2	3	4	5	6	7	8	9	10
A	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯
B	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯
C	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯
D	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯
E	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯

B Essay Questions (25 marks)

B.1 One Defective Ball ($5 + 4 = 9$ marks, plus 2 bonus marks)

Recall the question from Tutorial 4, where you had to find one heavier ball from a group of 243 balls. This question generalizes the problem. Suppose now we have n balls, and one of them is defective (either heavier **or lighter** than other $n - 1$ balls). Assume $n \geq 10$.

You have a balance, which allows you to check when given two sets A, B of balls (placed on either side of the balance respectively), whether:

1. A and B weigh equally, or
2. A is heavier than B, or
3. A is lighter than B.

Using the above balance, you need to devise an algorithm to detect which ball is defective and also whether it is heavier or lighter than other balls. As each weighing is charged, your algorithm needs to do it in as small a number of weighings as possible (in the worst case, in terms of n). For this problem, you need to give an optimal answer rather than in Θ notation. Your marks depend on whether your algorithm correctly finds the defective ball and how close your answer is to the optimal, though you are only required to prove upper bound (not lower bound).

Though you are allowed to use any method to solve the above question, the following parts guide you toward the answer in a structured/simpler way. (**Note:** If you decide not to follow the following guided path, you may use all the boxes for your own method; just specify at the top that you are not following the guided path.)

Remark: For the questions below, you should clearly specify your algorithm for how you are selecting the balls to be put on balance (and which side of the balance you place them on). However, you are not required to give any pseudo code.

a). (5 marks) Suppose you are given m balls, each colored RED or `green`. Among these balls, one is defective. Also you are also given one additional non-defective ball. Suppose you already know that:

- If the defective ball is RED, then it must be heavier than the other balls, and
- If the defective ball is `green`, then it must be lighter than the other balls.

What is the largest value of m, such that you can determine the defective ball among the m colored balls, in the worst case, using at most i weighings?

Hint: In each round, think about how many (and which) balls you want to put on the balance (and on which side of the balance).

For this scenario, we can go as large as $m = 3^i$ colored balls (plus one extra non-defective ball) to be able to correctly find the the defective (heavier or lighter) ball in i weighings. This is like what we discussed in Tutorial-4, but with some adjustments. Note that once we know a defective ball, we know whether it is heavier or lighter due to its color.

Base Case: In 0-weighings, one can determine the defective ball among 1 colored ball (trivial).

Induction: Suppose $i > 0$. If there are only two colored balls, then compare one of the balls with the given non-defective ball. If it is not the same weight as the non-defective ball, then it is the defective one. Otherwise, the one other remaining ball is defective.

If there are at least 3 colored balls, divide them in three (almost) equal groups A, B, C (each containing $\lfloor m/3 \rfloor$ or $\lceil m/3 \rceil$) balls) such that: group A and B have equal number of RED (respectively, green) balls. Note that this is always possible, as if RED (respectively, green) balls are odd in number, then the at most two odd balls (either from RED only, green only, or from both) can always be put in the group C. When we compare A and B, if they are equal weight, then group C has the defective ball. Otherwise, the RED ball in the heavier side or the green ball in the lighter side may be defective. Note that this reduces the number of potential defective balls to $\lceil m/3 \rceil$. Thus, by induction we can determine defective ball among 3^i colored balls using i weighings.

Note that this cannot be improved as in i trials one can have at most 3^i possible answers (as there are three results of each weighing), and thus we can only find a defective ball among at most 3^i balls.

b). (4 marks) Suppose you have n balls, where you have no other information (i.e., they are *not colored* as in part a).). Among these balls one is defective. You still have that extra one additional non-defective ball. Now, what is the largest value of n, such that you can determine the defective ball among the n balls (as well as whether it is heavier **or lighter** than the other balls), in the worst case, using at most i weighings?

Hint: In each round, think how many balls you want to put on the balance (and on which side).

Let $T(i)$ denote the largest number of balls from which you can find the defective ball using i weighings (in the worst case). Then try to express $T(i)$ in terms of $T(i-1)$ and result from part a), and then solve for $T(i)$.

We compute $T(i)$ as follows, for $i > 1$. Place 3^{i-1} balls plus the non-defective ball (so the total is even) in the balance (equally divide this even number of balls). If both sides are equal, then we have subproblem $T(i-1)$, i.e., the remaining $n - 3^{i-1}$ balls. In case one side is heavy, we color the heavier side as RED and the lighter side as green. There will be $m = 3^{i-1}$ colored balls now and using part a). solution, we can find the defective (heavy or light) ball in $i-1$ more weighings.

Thus, we get $T(i) = T(i-1) + 3^{i-1}$. Note that $T(1) = 1$ (compare this one ball with the non-defective ball). Thus, $n = T(i) = \sum_{j=0}^{i-1} 3^j = \frac{3^i-1}{2}$ (this is our answer).

Note that above is optimal as in i weighings we can have at most 3^i answers, which have to contain at least $2n$ possible solutions (that is n balls, each being heavier or lighter). As $2n$ is not a power of 3, we get 3^i must be at least $2n+1$, which matches the above upper bound for i weighings.

Further note: So if we do at most 1/2/3/4/5/6/7/i weighings, the largest n is 1/4/13/40/121/364/1093/$T(i)$, respectively.

Derivation of the sum done above: This is a sum of geometric progression series with $a = 3^0 = 1$, $r = 3$, and $x = i$ (as we sum from $j = 0$ to $i-1$, there are i terms), so $S_x = \frac{a \cdot (r^x - 1)}{(r-1)} = \frac{1 \cdot (3^i - 1)}{(3-1)} = \frac{3^i-1}{2}$.

c). **(Bonus 2 marks)** Now modify your algorithm and analysis for part b). when you are *not given* the additional non-defective ball (solving this variant will then solve the original question), to

determine the number of weighings needed to find the defective ball among the n balls (where $n \geq 10$).

Note that the only time we do not have the additional non-defective ball is when we do the weighing for the first time (after that, we always have at least one ball which is known to be non-defective — either due to the first balancing giving equal and thus any of these balls being weighed in the first balancing are non-defective (and you can use any); otherwise any of the remaining balls is the non-defective one). So, in the first round we only put $3^{i-1} - 1$ ball to make it equal number of balls in the two sides. Then after obtaining one ball that is guaranteed to be non-defective, the rest of the proof can go ahead as in part b).

Thus, we get $T(i) = \frac{3^i-1}{2} - 1 = \frac{3^i-1}{2} - \frac{2}{2} = \frac{3^i-3}{2}$. In other words, number of weighings needed for n balls (in the worst case) is: $\lceil log_3(2n+3) \rceil$.

*Here, we have assumed that $n \geq 3$. If $n = 2$, we cannot determine which ball is defective and if $n = 1$, we cannot determine if the ball is heavy or light.

Note again that above would be optimal, as in the first round we have to put equal number of balls in the two sides of the balance, and then we can follow the bounds of parts a). and b).

Further note: So if we do at most 1*/2/3/4/5/6/7/i weighings, the largest n is 0*/3/12/39/120/363/1092/$T(i)$, respectively.

Grading remarks by Prof Sanjay Jain below:

B1:

First note that if we are not after optimal number of weighings, but satisfied with approximate answer (with a constant additive factor for number of weighings), then below is a very simple solution for the question. Assume $n \geq 10$ as given in the question.

1. Divide the number n of balls into three groups A, B, C of nearly equal size (where A, B have size $\lfloor n/3 \rfloor$ and C has the remaining).
2. Compare A, B.

2.1. If A and B are equal, then compare C with equal number of balls from A, B (and we immediately know that C has the defective ball, and whether it is heavier or lighter).

2.2. If A and B are unequal, suppose A was heavier than B. Weigh A against $\lfloor n/3 \rfloor$ balls in C (note C contains only non-defective good balls). If A turns out to be heavier, then we immediately know that A has the defective ball, which is heavier. If weight of A is equal to C, then the defective ball is in B and must be lighter.

3. In any case, we have reduced the problem to at most $n - 2\lfloor n/3 \rfloor$ and used 2 weighings. Now use tutorial 4 trick to find the defective ball in the defective group using at most additional $\lceil \log_3(n - 2\lfloor n/3 \rfloor) \rceil$ weighings.

This is basically within additive factor of 1 to the optimal.

Thus, it is important to consider how close your answer is to the optimal

Following is the marking scheme for partial credit:

Maximum possible credit if you do perfectly with a weaker than optimal answer (note that this max applies only if your algorithm, answer and proof are perfect for the respective parts):

A. If your answer is $\log_3 n+$ constant: 4+4+1 for the three parts.

B. If your answer is $\log_2 n+$ constant: 3+2+1 for the three parts.

C. If your answer is $2\log_3 n+$ constant: 3+2+1 for the three parts, but usually a bit stricter.

D. If your answer is $\log_k n+$ constant, where $k < 2$ (other than the $2\log_3 n$ case above): 2+1+1 for the three parts

E. Linear and such cases: 1+1+1 for the three cases.

F. Some non-trivial tries: 1–2 overall (blank gets you 1/2+1/2).

If your answer is as above, but your argument or equations are not correct, there will deduction of marks. This really depends on how much you really achieved. Note that just guessing answer is not enough, your argument needs to justify the guess.

Penalty (for each part):

1 point if missing equation, incorrect equation based on argument (specially giving equation as 3^i when you cannot prove), or arguments has some issues.

In some cases manual deduction of further point if major/or lot of issues with the argument/equation. Usually I comment on the writeup in these cases. Note that without reasonable argument, you get no points.

In some cases, penalty is only given for only one of the part (a)/part (b), based on how much "problem" equation/argument has. Or penalty might be only 1/2 point for the relevant part.

Division in groups: many students didn't take care of odd number (when dividing in two groups) or divisible by 3 if dividing in 3 groups. Usually I didn't deduct points for this, but this depends on whether there are other issues with algorithm/equation. Note that there are cases when floor/ceiling could push the number of weighings needed by 1 (or based on the way some students argued, it could even double). So this is important to note.

Many students didn't attempt B1c, even though they could have gotten 1 point there as their earlier algorithm would work. Sorry, cannot help. Similarly, if you don't do a part, you don't get any points there. For those choosing their own method, I basically did the distribution of marks based on the scheme as above for their answer.

Afternote: For those who got $\log_3 n+$ constant as the answer, assuming their writeup for part (b) was near perfect, I gave them bonus for part (a): as they have almost got it right, and could have chosen to have taken "their own method". Sorry, this applies only if their part (b) equivalent basically had no errors.

B.2 Average Number of Inversions ($3 + 5 = 8$ marks)

Consider a sequence $a_1, a_2, \cdots, a_n$ of n distinct integers, where each $a_i \in \{1, 2, \cdots, n\}$. We call a pair (a_i, a_j) *inverted* if $i < j$ but $a_i > a_j$. For instance, the sequence $2, 3, 8, 5, 4$ contains three inverted pairs $(8, 5), (8, 4)$ and $(5, 4)$.

Let π be a permutation of $\{1, 2, \cdots, n\}$ chosen uniformly at random from the set of all permutations of $\{1, 2, \cdots, n\}$. Visualize this permutation π as a sequence $a_1, a_2, \cdots, a_n$ of n distinct integers.

a). Fix two integers $i, j \in \{1, 2, \cdots, n\}$. What is the probability that (a_i, a_j) forms an inverted pair? (Provide an answer with a proper explanation.)

Without loss of generality, assume $i < j$. Now consider the pair of elements (a_i, a_j). Observe that among all possible $n!$ permutations, exactly in half of them $a_i < a_j$ and in the remaining half $a_i > a_j$. To see this, let S be the set of all permutations where $a_i < a_j$ and L be the set of all permutations where $a_i > a_j$. Now, consider any arbitrary permutation $\sigma \in S$. Observe that swapping i-th and j-th element of σ leads to a distinct permutation in L. So essentially, there is a bijection between the sets S and L. Hence, $|S| = |L| = n!/2$.

Since π is chosen uniformly at random from the set of all permutations of $\{1, 2, \cdots, n\}$, the probability that (a_i, a_j) forms an inverted pair is $p = \frac{1}{2}$.

b). Suppose you are given a sorting algorithm that, given as input a permutation π with k inverted pairs, runs in $\Theta(k)$ time. Analyze the average-case complexity of this sorting algorithm. More specifically, what is the expected running time of this sorting algorithm when provided with as input a permutation of $\{1, 2, \cdots, n\}$ chosen uniformly at random from the set of all permutations of $\{1, 2, \cdots, n\}$? (Express the expected running time as a function of n using $\Theta(\cdot)$ notation.)

Note: If you cannot answer part (a), you may still answer this question (without any further penalty) assuming the answer to part (a) is p and then express the expected running time as a function of n, p using $\Theta(\cdot)$ notation.

For each i, j where $i < j$, let X_{ij} be the indicator random variable denoting whether $a_i > a_j$ or not, i.e.,

$$X_{ij} = \begin{cases} 1 \text{ if } a_i > a_j \\ 0 \text{ otherwise} \end{cases}$$

From part (a), we get that $\Pr[X_{ij} = 1] = 1/2$, and thus $\mathbb{E}[X_{ij}] = 1/2$.

Let X be the random variable denoting the number of inverted pairs in a randomly picked permutation. So $X = \sum_{(i,j):i<j} X_{ij}$. Then, by the linearity of expectation,

$$\mathbb{E}[X] = \sum_{(i,j):i<j} \mathbb{E}[X_{ij}] = \frac{n(n-1)}{4}.$$

Thus, the expected running time of the given sorting algorithm is $\Theta(n^2)$.

Note: This captures the behavior of insertion sort. One may observe that the running time of insertion sort is essentially $\Theta(n+k)$, where k is the number of inverted pairs of the input permutation. Hence, the average-case performance (expected running time when the input is chosen uniformly at random from the set of all permutations of n distinct elements) of insertion sort is $\Theta(n^2)$.

.

B.3 Stock Market ($5 + 2 + 1 = 8$ marks)

You have monitored the stock price of a certain popular company that you really like for the last N days: $P_1, P_2, \cdots, P_N$ (every P_i satisfies $1 \leq P_i \leq 777$ SGD). The stockbroker charges you a fixed rate

of C ($0 \leq C \leq 30$) for every stock purchase (there is no cost/fee when you sell your stock to the stockbroker). Today is day $N+1$; you wondered how much money you would have made if you had invested optimally during the previous day $[1..N]$. You are very risk averse so that at any given time in day $[1..N]$, you will never have more than one stock of the company. Note that there is a possibility that the answer is 0 SGD, i.e., it is best not to invest at all. Also, note that at the end, you need not hold any stock.

Example 1: If $N = 6$, $C = 10$ and $P = \{\mathbf{100}, 120, \underline{130}, 80, 50, 40\}$, the optimal strategy is to **buy one stock at day** 1**, paying** 100+10 = 110 **SGD**, wait, and on day 3, sell that one stock at 130 SGD. This way, you earn a profit of $130 - 110 = 20$ SGD. Do not do anything else on day $4, 5, 6$ (as the stock price of that company keeps declining).

Example 2: If $N = 6$, $C = 10$ and $P = \{100, 99, 99, 98, 97, 95\}$, the optimal strategy is to not do anything in these 6 days as the stock prices are non-increasing.

a). Describe the recurrence of this problem: the base case(s) and the general/recursive case(s)!

Let $profit(h, d)$ be the maximum profit at day d (the recurrence below assumes we process the days in this order $[0..d]$, but it is fine to do this in $[d..N-1]$ order too) if we have one stock ($h = 1$) or not ($h = 0$). Many students did not use this additional parameter h and complicates the DP solution. For $\Theta(N)$ full-solution, one should use this Boolean parameter h (otherwise there is a need to do an $O(n)$ loop to compute each DP state and then we cannot get $\Theta(N)$ overall complexity).

Then the only base case is $profit(h, N) = 0$ — no more day to process

There are three subcases (but only two subcases are possible per day) to process:
$profit(h, d) = profit(h, d+1)$ — skip day d (always possible)
$profit(h, d) = P[d] + profit(0, d+1)$ — has one stock ($h = 1$) and sells one stock at day d
$profit(h, d) = -P[d] - C + profit(1, d+1)$ — has no stock ($h = 0$) and buys one at day d
A substantial number of students write recurrences that involves all three at the same day.

b). 1). What is the running time of the recurrence above if you implement your recursion verbatim?
2). What is the *faster* running time if you implement the recurrence using Top-Down DP (recursion *with memoization*) or Bottom-Up DP? (Express **both** running times using $\Theta(\cdot)$ notation.)

There are 2 branching options per day depending whether you own one stock or not and there are d days, so we have $\Theta(2^N)$ (1 mark).

If we use Top-Down DP (or Bottom-Up DP), we will only compute $2 \times N$ distinct subproblems: each of the N days has 2 states: own one stock or not. Each of these subproblems can be computed in $\Theta(1)$. Therefore the overall time complexity is $\Theta(N)$ (1 mark).

A substantial number of students write an alternative solution that is $O(N^2)$; marks deducted in section B3a). for this as $O(N^2)$ is substantially slower than $\Theta(N)$ solution.

c). Using the DP solution that you have provided above, be a human computer and compute what is the optimal answer (just give a single integer) if you are given $N = 20$, $C = 30$, and $P = \{10, 80, 20, 40, 30, 50, 40, 60, 50, 70, 60, 10, 200, 199, 198, 197, 196, 195, 194, 193\}$.

The last 7 values are not good as it keeps decreasing, so we should sell stock when it's price is at the highest 200 SGD. We need to own one stock before day $d = 13$ to do this, etc... Filling this backwards, we should arrive at 220 SGD profit. We **buy** and sell at the following days: $P = \{\mathbf{10}, 80, \mathbf{20}, 40, 30, 50, 40, 60, 50, 70, 60, \mathbf{10}, 200, \text{[ignore the rest]}\}$. We get $(80 - 10 - 30) + (70 - 20 - 30) + (200 - 10 - 30) = 40 + 20 + 160 = 220$ SGD (admittedly tricky to compute, so only do this when you have enough time nearing the end of the paper to do this, as leaving this box blank - 0.5 mark = is more beneficial than rushing the computation and getting it wrong - 0 mark).

Notice we do not do anything on day 11 (60 SGD) - and also from day 14 to 20. Some recurrences that students proposed failed to into account this possibility of skipping some day(s) (e.g., if you insist to buy at day 11 and immediately sell it again, you lose $-C$ dollars if $C > 0$, not optimal; same if you insist to do anything in days 14..20).

– END OF PAPER; All the Best –