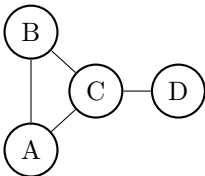


Statistics (N = 236)

Full mark = 45. Maximum = 42. Median = 13. Advance to Final = 15.5 marks or above.

Section A

Q	A	Explanation
1	A	<p>When adding two characters in C++, the result will be the sum of their ASCII value, so <code>one + zero</code> will be 97, the program then stores 97 in a character variable, so <code>ten</code> will be the character with ASCII value 97, which is 'a'.</p> <p>Consider the figure below.</p> 
2	B	<ul style="list-style-type: none"> • There are 3 possible colours for node <i>C</i>. • There are 2 possible colours for node <i>B</i>, except the one colour chosen for node <i>C</i>. • There is 1 possible colour for node <i>A</i>, except the two colours chosen for node <i>B</i> and <i>C</i>. • There are 2 possible colours for node <i>D</i>, except the one colour chosen for node <i>C</i>. <p>Hence, there are in total $3 \times 2 \times 1 \times 2 = 12$ ways of colouring.</p>
3	B	<p>The hyphen symbol ('-') is the negative / subtract operator in C++, it cannot be used in a variable name.</p> <p>A variable can start with an underscore symbol ('_'), and only such symbol is a valid variable name. <code>define</code> is a token with meaning to the preprocessor, but not a reserved keyword in C++, so it may still be used as a variable name.</p>
4	B	<p>The outer layer loops <i>i</i> through all values from 0 to 10 (11 values), and the inner layer loops <i>j</i> through all values from 0 to 9 (10 values). Meaning that the if-statement has gone through all 110 possible pairs of <i>i</i> and <i>j</i>.</p> <p>Since <code>cnt</code> is incremented only when both <i>i</i> and <i>j</i> are even, we can ignore cases where <i>i</i> is odd or <i>j</i> is odd.</p> <p>There are 6 even numbers from 0 to 10 and 5 even numbers from 0 to 9, so the answer is $6 \times 5 = 30$.</p>

Q	A	Explanation
		<p>We could estimate the operation runtime by estimating the number of instructions needed. Usually, we represent it with the Big-O notation.</p> <p>Consider the 3 operations.</p> <p>i. Searching for an element in a linked list has a worst-case time complexity of $O(n)$ as a linked list does not support random access and we need to loop through the whole list.</p> <p>ii. Prepending an element to a linked list is always $O(1)$ as we can simply modify the head pointer.</p> <p>iii. Bubble sort has a worst-case time complexity of $O(n^2)$, the use of a linked list will not bring any speed-ups to the algorithm so it is still $O(n^2)$.</p> <p>Hence we can estimate the order to be ii, i, iii</p>
6	B	<p>We start with $((\text{NOT } C) \text{ AND } (B \text{ OR } (\text{NOT } B))) \text{ OR } (A \text{ AND } C)$.</p> <p>Apply the negation law to get $(B \text{ OR } (\text{NOT } B)) = \text{TRUE}$.</p> <p>Then apply the identity law to get $((\text{NOT } C) \text{ AND } \text{TRUE}) = \text{NOT } C$.</p> <p>Now we are left with $(\text{NOT } C) \text{ OR } (A \text{ AND } C)$</p> <p>Apply the distributive property of OR to get $((\text{NOT } C) \text{ OR } A) \text{ AND } ((\text{NOT } C) \text{ OR } C)$.</p> <p>Once again apply negation law and identity law to get $(\text{NOT } C) \text{ OR } A$</p>
7	D	<p>Simply dry run the code, we can find the answer is 14.</p>
8	A	<p>i. True.</p> <p>ii. True. Merge Sort recursively splits the array into two halves, sorts them individually, and merges them back together to achieve a sorted array.</p> <p>iii. False. Selection sort works by repeatedly selecting the smallest (or largest) element from the unsorted part of the array and putting it in front (or back). The sorting method described in the question is quicksort.</p>
9	B	<p><code>s[0]='\t', s[1]='\\', s[2]='\t', s[3]='\\', s[4]='\t', s[5]='\\', s[6]='\\', s[7]='\t'</code> where <code>'\t'</code> is the tab character (ASCII Code 9) and <code>'\\'</code> is the backslash character (ASCII Code 92). A backslash escapes the character behind it.</p>
10	A	<p>i. True. This is the contrapositive of the first statement.</p> <p>ii. False. Consider the contrapositive of the second statement “If Bob is not unhappy, then it is not (raining outside and today is a Monday)”.</p> <p>We can apply De Morgan’s theorem to get “If Bob is not unhappy, then it is not raining outside or today is not a Monday”.</p> <p>Hence Bob can be not unhappy while raining outside and today not being a Monday.</p> <p>iii. False. The event “Alice brings an umbrella” does not mean anything as she could have brought it because of other reasons. Here it is important to note the direction of implication cannot be reversed.</p> <p>iv. False. Similar to iii. The event “Bob is unhappy” does not mean anything as he could have been unhappy because of other reasons. Here it is also important to note the direction of implication cannot be reversed.</p> <p>Hence, only i is correct.</p>

Q	A	Explanation
11	D	i. False. One counterexample is $a = 3$. In this case, $b = 1$ and $c = 2$, thus $a = c$ is false.
		ii. False. One counterexample is $a = 0$. In this case, $b = 0$ and thus $a > b$ is false.
		iii. False. One counterexample is $a = 0$. In this case, $b = 0$ and $c = 0$, thus $b \neq c$ is false.
		Hence, none of them must be true.
12	B	The number of ways to form an integer with n digit(s) with 1, 3 and 7 are 3^n . As $3 + 9 + 27 \leq 55 \leq 3 + 9 + 27 + 81$, the 55^{th} smallest positive integer consists of 4 digits. Then the problem becomes finding the $55 - 3 - 9 - 27 = 16^{\text{th}}$ 4 digits smallest positive integer. Convert 15 into base 3, we have $15_{10} = 120_3$. By substituting 1, 3, 7 into 0, 1, 2 respectively, the answer is 1371.
13	D	First of all, we know that $f(0, x) = x$ and $f(x, 0) = x$. Calculate the values of all other possible calls of $f(x, y)$: $f(3, 4) = f(3, 2) + f(1, 4)$, $f(3, 2) = f(1, 2) + f(3, 1)$, $f(1, 4) = f(0, 4) + f(1, 2)$, $f(1, 2) = f(1, 1) + f(0, 2)$, $f(3, 1) = f(3, 0) + f(1, 1)$, $f(1, 1) = f(1, 0) + f(0, 1)$. From above we know that $f(1, 1) = 2$, $f(3, 1) = 5$, $f(1, 2) = 4$, $f(1, 4) = 8$, $f(3, 2) = 9$, and finally $f(3, 4) = 17$.
14	B	In machine learning, the training process involves providing a model with a dataset containing examples and allowing the model to learn patterns, rules, and strategies from this data. For AlphaCode, public GitHub repositories and contest submissions are used in the training process. Watch this video to learn more about the underlying mechanism of AlphaCode.
15	B	While $n = 4k + 3$ for some integer $k \geq 0$, we have $n \% 4 == 3$, after one iteration the new value is $n = 4k - 1$, i.e. n decreases by 4. Starting with $n = 2023$, it keeps decreasing by 4 until $n = -1$, where then $-1 \% 4 == -1$, and the while condition is no longer satisfied and the loop terminates.
16	C	We consider the number of inversions in <code>arr</code> , which is defined as the number of pairs of indices (i, j) such that $i < j$ and <code>arr[i] > arr[j]</code> . In the initial array <code>[3, 2, 6, 4, 1, 5]</code> , there are 7 inversions. We also note that each swap changes the number of inversions by either +1 or -1, thus changing the parity. This is because when we swap indices $(i, i + 1)$, for the pair $(i, i + 1)$, if it was not an inversion it will become one and vice versa, and for all pairs except $(i, i + 1)$ we can simply replace all i with $i + 1$, $i + 1$ with i , keeping the count the same. Therefore the total number of inversions change by either +1 or -1. When <code>arr</code> is sorted in ascending order, it has 0 inversions, and when <code>arr</code> is sorted in descending order, it has $\frac{6 \times 5}{2} = 15$ inversions. Considering the parity of the number of inversions, only ii and iii are true.
17	A	Note that both loops use the same variable i . Keep track of its value carefully.
18	D	As soon as a "Tails" is flipped, Bob will win. $P(\text{Alice wins}) = P(HHT) + P(HHHT) + P(HHHHT) + \dots = \frac{1}{2^3} + \frac{1}{2^4} + \frac{1}{2^5} + \dots = \frac{\frac{1}{2^3}}{(1 - \frac{1}{2})} = \frac{1}{4}$ $P(\text{Bob wins}) = \frac{3}{4}$

Q	A	Explanation
		<p>Let a, b, c, d, e, f, g be 0 or 1 values denoting whether the corresponding nodes are safe.</p> <p>The deduction can be done as follows:</p>
		<p>1. From the information in nodes A and B, we know that $a + b + c + d = 3$ and $a + b = 1$. Hence, $c + d = 2$, i.e. C and D are both safe.</p>
19	A	<p>2. From the information in nodes E, F, G, we know that $d + e + f + g = 3$, $e + f = 1$ and $e + g = 1$. Hence, $d + g = 2$ and $d + f = 2$, i.e. D, F, G are all safe.</p> <p>3. Substitute it back to $d + e + f + g = 3$, we know that E is not safe.</p>
		<p>4. Both $a = 1, b = 0$ and $a = 0, b = 1$ results in a valid configuration that satisfies all constraints. Therefore, we do not have sufficient information to determine whether a or b is safe or not.</p>
20	A	<p><code>f()</code> is a recursive function which will change the value of <code>x</code> in <code>main()</code> since <code>x</code> is a global variable. Note that there is no <code>return</code> inside the second if-statement, so the program will proceed to the third if-statement after the execution of the second if-statement with the changes to <code>x</code>.</p>
21	D	<p>The inner loop can be simplified into <code>tmp = b * ans</code>, so each iteration of the outer loop is performing the operation of <code>ans += b * ans</code>, or equivalently <code>ans *= (1 + b)</code>. Therefore, the program calculates $ans \times (1 + b)^a = 1 \times (1 + 3)^5 = 1024$.</p>
22	D	<p>The outer loop is performing binary search on array <code>a</code> using bitwise operations. The inner loop is computing the prefix sum from 0 to $x - 1$ in every iteration of the outer loop. Overall, the program finds the first element of <code>a</code> such that the prefix sum up to that element is equal to or greater than 79. The prefix sum up to 5-th element (0-based) is exactly 79, and its value is 29.</p>
23	B	<p>In <code>f()</code>, the first argument <code>a</code> is passed by reference while the second argument <code>b</code> is passed by value. It means that any changes made to <code>a</code> in <code>f()</code> will affect its value outside <code>f()</code>, while any changes made to <code>b</code> in <code>f()</code> will not affect its value outside <code>f()</code>.</p> <p>In <code>main()</code>, <code>b</code> and <code>a</code> are passed to <code>f</code> as the first and second argument respectively, so the value of <code>b</code> in <code>main()</code> is updated after executing <code>f(b, a)</code>. The new value of <code>b</code> is the same as the return value of <code>f(b, a)</code>, which is $2 + 1 + 2 = 5$.</p> <p>The value of <code>a</code> after <code>a = f(b, a) + a</code> is $5 + 1 = 6$. Hence, the final value of <code>b</code> is $5 + 6 = 11$.</p>

Q	A	Explanation
		Notice that for each iteration, the code splits the range by half, it can be expressed as a tree as follows.
24	B	<p>For the i^{th} level, the range is split into 2^i parts, each size is 2^{4-i}. Therefore summing up the square of all nodes is $1 \times 16 \times 16 + 2 \times 8 \times 8 + 4 \times 4 \times 4 + 8 \times 2 \times 2 + 16 \times 1 \times 1 = 496$</p>
		It will be tedious to enumerate all subsequences, consider a more efficient method by dynamic programming: let $f[i][j]$ be the number of ways to form the numbers having a remainder of j being divided by 6 considering the first i digits in the sequence. We then have the following transition formula: Let $a[i]$ be the i^{th} digits in the sequence, then for each i, j :
25	A	<pre>f[i][a[i]] += 1 // Contribution from the current digit alone f[i + 1][(j * 10 + a[i + 1]) % 6] += f[i][j] // Using the next digit f[i + 1][j] += f[i][j] // Not using the next digit</pre> <p>The answer is then $f[7][2] + f[7][4]$, which equals to $18 + 18 = 36$. In fact, the answer also equals to $f[6][2] + f[6][4]$ since any number ending with 3 would not be divisible by 2.</p>

Section B

Q	Answer and Explanation
A1	$xA * pA + xB * pB \leq N$ This function uses xA and xB to exhaust the number of gift A and B redeemed. This line checks if the total number of gift points used is less than or equal to N .
A2	$xA * vA + xB * vB$ It calculates the total value of xA gift As and xB gift Bs.
B	$xA * vA + (N - xA * pA) / pB * vB$ After redeeming xA gift As, Alice has $N - xA * pA$ gift points remaining, dividing it by pB calculates the maximum number of gift B redeemable using the remaining gift points.
C	$xA * vA + \text{FindMaxValueFaster}(N - xA * pA, pB, vB, pC, vC)$ // $xA * vA + \text{FindMaxValue}(N - xA * pA, pB, vB, pC, vC)$ $\text{FindMaxValueFaster}$ and FindMaxValue calculates the maximum value obtainable using the remaining gift points to redeem gift B and C.

Q	Answer and Explanation
D1	$s[i] - 'A' + 1 \text{ // } s[i] - 64 \text{ // } s[i] \% 65 + 1$ By subtracting the ASCII value of the character A and adding 1 back, we map each uppercase alphabet to the required value.
D2	$4 - i$ We have to raise the first character to its fourth power, second character to third power and etc. Which is exactly the reversed iterated value of i .
E	$PAFR \text{ // } PBEV \text{ // } PBFK \text{ // } PCBX \text{ // } PCCS \text{ // } PCDL \text{ // } PCEC$ $PAGE = 16^4 + 1^3 + 7^2 + 5 = 65536 + 1 + 49 + 5 = 65591$. Note that by fixing the first character, the remaining value is 55 which can be very small and therefore we can only choose small value for the remaining three characters.
F	$(2, 3, 6, 5, 2, 4) \text{ // } (2, 6, 3, 5, 2, 4) \text{ // } (3, 2, 6, 5, 2, 4) \text{ // } (6, 2, 3, 5, 2, 4) \text{ // } (2, 4, 2, 3, 5, 6)$ We can do exhaustion quickly by cutting some branches, observe that: (1) we must stop at the key, so the sum of steps before and after the key should be both 11. (2) there are many blockers at the end of the road, we could do exhaustion for the later part and check if the remaining steps could fit the first part.
G	1296 Let $P(i)$ be the number of ways that could arrive at i^{th} cell. Then the value of $P(i)$ is the sum of $P(i-6)$, $P(i-5)$, ..., $P(i-1)$ as we could walk 1 to 6 steps each time. Notice we should calculate the number of ways to reach the key and use that answer as a starting point to calculate $P(\text{last cell})$ again as we must stop at the key.
H1	$p[i] != -1 \text{ // } p[i] > -1 \text{ // } p[i] >= 0 \text{ // } \sim p[i]$ To delete a targeted element in a singly linked list which every node stores the next pointer, we need to check if the next element is the target, and then delete the target from current node by changing the current node's pointer pointing to the "next next" element. If it is the last node of the linked list, we cannot access the next element as $p[i] = -1$, so we need to handle this case to avoid accessing $a[-1]$.
H2	$a[p[i]]$ Check if the next element is the target after handling the corner case, as mentioned in H1.
H3	$p[i] = p[p[i]]$ Delete the target from current node by changing the current node's pointer pointing to the "next next" element, as mentioned in H1.
I	$= 3 \text{ // } \&6 \text{ // } \%7 \text{ // } , 3$ $a \% 7$ equals to 1, 2, 3, 4, 5, 6, 0 for every 7 numbers from 1 to 70. As the sum of the above equals 21 and the cycle loops 10 times, the answer is $21 \times 10 = 210$. In terms of base 2, $6 = 110_2$. $a \& 6$ keeps all 2^1 and 2^2 in a and set other bits to 0. From 1 to 70, there are 35 numbers with the bit $2^1 = 1$, and 35 numbers with the bit $2^2 = 1$. In total, the answer is $35 \times 2^1 + 35 \times 2^2 = 210$. $a = 3$ assigns 3 to a and return the value of a , which is 3. Comma operator discards the result of the first expression and return the result of the second expression, therefore $a, 3$ return 3 as well. The answers of both expressions above are $70 \times 3 = 210$.

Q	Answer and Explanation														
J	256														
	Observe that the value of $i \ \& \ -i$ is equal to the largest power of 2 that divides i , so we can group the numbers as follows:														
	<table><tr><th>Group</th><th>$i \ \& \ -i$</th><th>Numbers</th></tr><tr><td>1</td><td>1</td><td>1, 3, ..., 63</td></tr><tr><td>2</td><td>2</td><td>2, 6, ..., 62</td></tr><tr><td>...</td><td>...</td><td>...</td></tr><tr><td>6</td><td>64</td><td>64</td></tr></table>	Group	$i \ \& \ -i$	Numbers	1	1	1, 3, ..., 63	2	2	2, 6, ..., 62	6	64
Group	$i \ \& \ -i$	Numbers													
1	1	1, 3, ..., 63													
2	2	2, 6, ..., 62													
...													
6	64	64													
	So the final answer is $1 \times 32 + 2 \times 16 + 4 \times 8 + 8 \times 4 + 16 \times 2 + 32 \times 1 + 64 = 256$.														
K	<pre>n, n/i*i/2 // (n+1)/2, (n/i+1)/2*i // (n+1)/2, (n+i)/i/2*i // n-n/2, i*(n/i-n/(i*2))</pre>														
	For the first solution, all numbers from 1 to n have at least 1 contribution to the answer. There are $n/2$ numbers that have at least 2 contribution to the answer. As we have already added 1 for all numbers, we only need to add 1 to the answer for each of them. There are $n/4$ numbers that have at least 4 contribution to the answer. As we have already added $1 + 1 = 2$ for each of them, we only need to add 2 more. We can do the same for 8, 16, ..., up to the largest power of 2 that is smaller than or equal to n . For the second, third and fourth solution, there are $(n+1)/2$ numbers whose $i \ \& \ -i$ is 1, and $(n/i+1)/2$ numbers whose $i \ \& \ -i$ is 2, etc..														
L	23														
	The return type of <code>a.size()</code> is <code>size_t</code> , which is a 64-bit unsigned integer type. When it is equal to 0, <code>a.size() - 1</code> is equal to 2^{64} , which will cause an infinite loop in line 21 as <code>i</code> is a 32-bit signed integer. When <code>i</code> is out of the range of the vector, it will cause runtime error in line 23.														
M	<pre>21, i<n-1; // 21, i+1<n; // 21, i+1<a.size();</pre>														
	<code>n</code> is an 32-bit signed integer and its value is equal to <code>a.size()</code> . We can either use <code>n</code> instead of <code>a.size()</code> or move the <code>-1</code> to the left side to be <code>+1</code> to avoid the infinite-loop problem.														