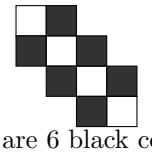
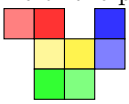
Hong Kong Olympiad in Informatics 2025/26
Heat Event (Senior Group)
Official Solution

Statistics (N = 450)
Full mark = 45. Maximum = 45. Median = 13.5. Advance to Final = 21.5 marks or above.

Paper 1 (Compulsory Part)

Section A

Q	A	Explanation
1	B	i. False. Notice that if a shape can be completely covered, it must have the same number of black and white cells when coloured in a checkerboard pattern.  There are 6 black cells and 4 white cells. ii. False. We can place the 1×2 tiles greedily, prioritizing cells with lower number of uncovered neighbours. We will always end up with cells with all neighbours covered by a tile but itself is not covered. iii. True. One of the possible ways is as follows: 
2	C	Only the statement about Merge Sort is fully correct. • Selection Sort works by repeatedly finding **the minimum (or maximum) element** from the unsorted section and moving it to the end of the sorted section. It does not select the median, nor does it split the array into halves. • Insertion Sort takes the first unsorted element and inserts it into its **correct** sorted position within the sorted section. It does not simply put it at the "frontmost" position every time (that would just reverse the list). • The description "compares adjacent elements quickly and repeatedly swaps them" describes **Bubble Sort**, not Quicksort. Quicksort works by selecting a pivot and partitioning the array.
3	B	Only Bob can use only his exam papers to disprove the teacher's claim. • To prove the teacher's claim, we need to check all students who got 50 marks or above in the Chinese exam, and ensure that they all got 50 marks or above in the English exam. Only checking one student who got 50 marks or above in the Chinese exam is not sufficient. • To disprove the claim, we only need to have one counter example, which is a student with 50 marks or above in the Chinese exam but got less than 50 marks in the English exam, which Bob did.

Q	A	Explanation
4	B	i. False. A graph with 5 nodes does not need to have any edges at all. It could be an empty graph consisting of 5 isolated vertices and 0 edges. The condition "at least 4 edges" would only be necessary if the graph were required to be **connected**. ii. True. In a simple undirected graph, the maximum number of edges occurs when every node is connected to every other node (a complete graph). The number of edges is calculated as $\frac{n(n-1)}{2}$. For $n = 5$, this is $\frac{5\times4}{2} = 10$. iii. False. A graph with a single node ($n = 1$) has 0 edges, but it is considered a connected graph because there is a path (of length 0) from the node to itself, and there are no disconnected parts.
5	B	Let M be the maximum value in the row. Consider the **first** occurrence of M from the left, denoted as x_i: • **Interesting:** Since x_i is the first instance of the maximum, all numbers to its left are strictly less than x_i. • **Exciting:** Since x_i is the maximum value, it is $\geq$ all numbers to its right. To prove uniqueness, suppose there are two such numbers x_a and x_b where $a < b$: 1. For x_b to be interesting, $x_b > x_a$. 2. For x_a to be exciting, $x_a \geq x_b$. Since these two conditions contradict each other ($x_b > x_a$ and $x_a \geq x_b$ cannot both be true), there can only be exactly one such number.
6	D	The first element popped from R is 3. Since 3 is pushed last into R, we can conclude that R is a first-in-last-out data structure, i.e. a stack. Then 2, 1 are popped from R and pushed into S in order. Since the next element popped from S is 1, we can conclude that S is also a first-in-last-out data structure, i.e. a stack.
7	A	i. True. Since `s` itself is a palindrome, `s` can be written as $ABCDDCBA$ where A, B, C, D each denote a character. Then, since `s[1..4]` is palindrome, $A = D$ and $B = C$, implying `s[5..8]` is palindrome as well. ii. False. A counterexample would be `AAABBAAA`.
8	B	i. False. When the initial value of `remain` is 10, `v[1]` and `v[2]` are both `False`, but `v[0]` is `True`. ii. True. If `v[4]` and `v[5]` are both true, then the value of `remain` before index 4 must be at least `p[4]` + `p[5]` $= 2 + 3 = 5$. Since `remain` $\geq 5 =$ `p[3]` at index 3, `v[3]` must be `True`.
9	A	The recursive structure follows the pattern `f(n)` = `f(n-1) n f(n-1)`, with `f(0)` producing no output. • `f(1)` = `1` • `f(2)` = `1 2 1` • `f(3)` = `1 2 1 3 1 2 1` • `f(4)` = `1 2 1 3 1 2 1 4 1 2 1 3 1 2 1`

Q	A	Explanation
10	D	First, since each route passes through exactly 6 edges, the average load of the edges is $\frac{6\times6}{12} = 3$. Therefore, the maximum load is at least 3, and it can only be attained if the load of all edges is exactly 3. However, we can show that this is impossible to attain. Without loss of generality suppose A is routed clockwise, and consider the first and last edge on the route as highlighted in Figure 1. The routes for the requests B to F will pass through exactly one of these two edges, so the total load of these edges will be $= 2+1+1+1+1+1 = 7$, which is > 6. Therefore, the maximum load is at least 4, which can be achieved by routing requests A, C, E clockwise and requests B, D, F anti-clockwise (as shown in Figure 2).

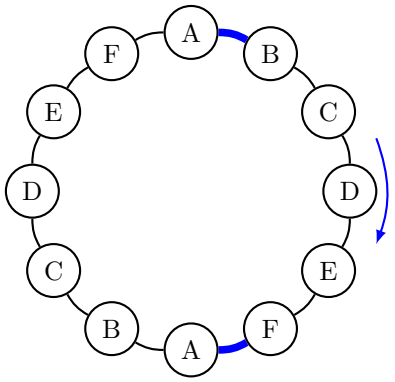


Figure 1: Maximum load = 3 is impossible

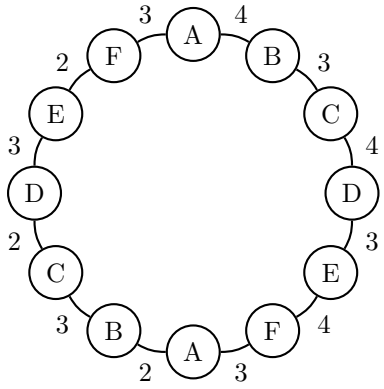


Figure 2: Maximum load = 4 is possible

Section B

Q	Answer and Explanation
A	`9`
	Consider working backwards from the final result. In order to reduce the number with the minimum number of clicks, it is optimal to divide by 3 whenever possible. $$330 \to 110 \to 109 \to 108 \to 36 \to 12 \to 4 \to 3 \to 1 \to 0$$ Therefore, at least 9 clicks are necessary.
B	`0,1,1`
	Initially, $a = b = c = 1$. The values of a, b, c after the first few iterations are listed as follows: i | 0 1 2 3 4 5 6 7 8 ... a | 0 0 1 1 1 0 1 0 0 ... b | 0 1 0 0 1 1 1 0 1 ... c | 1 1 0 1 0 0 1 1 1 ... Note that the values of a, b, c form a cycle of length 7. So, after the 2025-th iteration, the values of a, b, c are $0, 1, 1$ respectively.
C	`-28`

	Observe that we can split the for-loop into two as follows:
	`for (int i = 0; i < 56; ++i) ans += (i % 7);`
	and
	`for (int i = 0; i < 56; ++i) ans -= (i % 8);`
	Note that the values of `i % 7` and `i % 8` forms cycles of length 7 and 8 respectively. Therefore, the answer is $(0+1+\cdots+6)\times 8 - (0+1+\cdots+7)\times 7 = -28$.

Q	Answer and Explanation
D	`15`
	We first show that the necessary and sufficient condition is that the first 6 seats have exactly four occupied seats and any two seats at distance 6 apart have the same occupancy status. Number the seats $1, 2, \ldots, 12$ clockwise, and let a_i be 1 if seat i is occupied and 0 otherwise. Compare the two sets of six consecutive seats starting at seats i and $i+1$. Since both sets contain exactly four occupied seats, removing seat i and adding seat $i+6$ cannot change the number of occupied seats. Hence, $a_i = a_{i+6}$ for every i, so every pair of opposite seats has the same occupancy status. Also, the first six seats must contain exactly four occupied seats. Conversely, suppose these two conditions hold. Any six consecutive seats contain exactly one seat from each of the six pairs of opposite seats. Since the two seats in each pair have the same occupancy status, any six consecutive seats contain the same number of occupied seats as the first six seats, namely four. Thus, the conditions are sufficient. It remains to choose which four of the first six seats are occupied. The occupancy of the other six seats is then uniquely determined by that of their opposite seats. Therefore, the number of seating arrangements is $\binom{6}{4} = 15$.
E	`10`
	Let $P[i]$ represent the sum of the first i elements, $P = [0, 6, 6, -2, 2, 10, 16, 12, 14]$. The sum of subarray $A[L], A[L+1], \ldots, A[R]$ is equal to $P[R+1] - P[L]$. The answer is the number of pairs $(P[L], P[R+1])$ such that their difference is between 10 and 14, inclusive. There are a total of 10 valid pairs: $(P[0], P[5])$, $(P[0], P[7])$, $(P[0], P[8])$, $(P[1], P[6])$, $(P[2], P[6])$, $(P[3], P[5])$, $(P[3], P[7])$, $(P[4], P[6])$, $(P[4], P[7])$ and $(P[4], P[8])$.
F	`(a+b+c)%3`
	When a, b, c are all the same, the sum would be one of $3, 6, 9$. When they are all different, the sum would be 6. Otherwise, it would be two same one different (in the form x, x, y), giving sum $\equiv 1$ or $2 \pmod 3$ after listing out all possibilities of x and y.

Section C

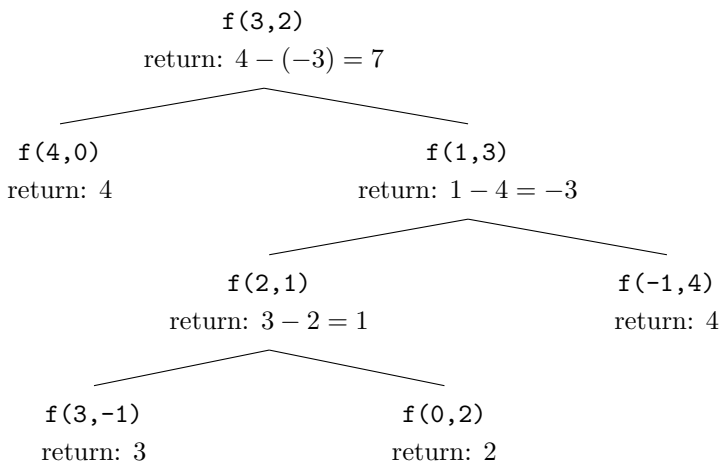
Q	Answer and Explanation	
G	G1	`3-j`
	G2	`i`
	In the i-th turn of the outer loop, we generate the i-th row of the output, which corresponds to the i-th column of the original matrix a, which means that the column index `G2` must be i. Within the nested loop, as j increases from 0 to 3, we need to extract elements from that column from the bottom to the top, so the row index `G1` must start at 3 and decrease as j increase, resulting in the formula $3 - j$.	
H	H1	`i//2` // `i/2`
	H2	`j%2*2`
	For the row index `H1`, we notice that in the first two output rows, the first index needs to be 0 to access the first row of array a, while for the next two output rows, the first index needs to be 1. Therefore, we use `i/2` in C++ and `i//2` in Python to keep the index constant for every two lines. For the column index `H2`, the value on even-numbered columns should be 0 and the value on odd-numbered columns should be 2. We can use `j%2` to group the columns by parity and then use `j%2*2` to locate the corresponding column in array a.	
I	`0`	
	The shortest walk from A to E is $A \to B \to C \to D \to E$ of length 4. Any other longer walks have a length ≥ 6. Therefore, it is impossible to walk from A to E with exactly 5 edges and the answer is 0.	
J	`2`	
	There are two disjoint walks from A to E: $A \to B \to A \to B \to C \to D \to E$ and $A \to F \to I \to M \to L \to J \to E$. Therefore, we have to delete at least 2 edges. One possible way is to delete the edges $D \to E$ and $J \to E$, as this cuts off all paths from A to E.	
K	`1`	
	Note that 9 is odd. The only way to walk from A to E using an odd number of edges is by passing through the edge $K \to M$. Once we delete the edge $K \to M$, the graph becomes *bipartite*, i.e. we can colour the vertices black or white such that each edge connects a black vertex and a white vertex. Since A and E share the same colour, all walks from A to E will have even length, and there are no walks of length 9.	

Paper 2 (Python)

Section A

Q	A	Explanation
1	D	The program calculates the sum of all elements in the list `a`. The total sum is: $1+2+3+4+5+6+7+8+9+10 = \frac{(1+10)\times 10}{2} = 55$
2	A	The expression `a[i], a[3 - i] = a[3 - i], a[i]` swaps the values at indices i and $3-i$. When $i = 0$, `a[0]` and `a[3]` are swapped; however, when $i = 3$, these same two elements are swapped again, which restores them to their original values. Similarly, when $i = 1$ and $i = 2$, the values of `a[1]` and `a[2]` are swapped twice. Because every pair within the loop range is swapped an even number of times, the array a remains unchanged. Therefore, the final output is the initial value of the array: 32768.
3	A	$m = 20$, so x is uniform in $\{0, 1, \ldots, 19\}$. • When $x < 14$ (14 values), gain is $+1$ • When $14 \le x < 17$ (3 values), gain is $+5$ • When $17 \le x < 19$ (2 values), gain is $+10$ • When $19 \le x < 20$ (1 value), gain is $+100$ Therefore expected gain per round would be $\frac{14}{20}\times 1 + \frac{3}{20}\times 5 + \frac{2}{20}\times 10 + \frac{1}{20}\times 100 = 7.45$. By linearity of expectation, expected gain after $n = 10$ rounds would be $7.45 \times 10 = 74.5$, so the answer is A.
4	B	Note that Python passes arguments by object reference. Integers are immutable, therefore the local assignment of `x` does not affect `a[0]`. However, lists are mutable, and `y` refers to the same list as `a`, so the assignment `y[1] = 1` mutates this list. Therefore, only `a[1]` is changed.
5	A	• Initially, `a = [-1, -1, -1, -1, -1, -1, -1]` (Size 7) • `p(8)`: Target index $8 \pmod 7 = 1$. Slot 1 is empty. Insert 8 at index 1. • `p(15)`: Target index $15 \pmod 7 = 1$. Slot 1 occupied. Check next slot (2). Empty. Insert 15 at index 2. • `p(3)`: Target index $3 \pmod 7 = 3$. Slot 3 is empty. Insert 3 at index 3. • `p(1)`: Target index $1 \pmod 7 = 1$. Slots 1, 2, 3 occupied. Check next slot (4). Empty. Insert 1 at index 4. • `p(10)`: Target index $10 \pmod 7 = 3$. Slots 3, 4 occupied. Check next slot (5). Empty. Insert 10 at index 5. **Final Array:** `[-1, 8, 15, 3, 1, 10, -1]`

Section B

<table>
<tr><th>Q</th><th>Answer and Explanation</th></tr>
<tr><td>A</td><td><code>1</code></td></tr>
<tr><td></td><td>The values of <code>x</code> are 1, 0, 1, 0, 1, 1 after each iteration.</td></tr>
<tr><td>B</td><td><code>7</code></td></tr>
<tr><td></td><td>The recursion tree, as well as the return value for each of the calls, is shown below.
 

So, the output is 7.</td></tr>
<tr><td>C</td><td><code>0</code></td></tr>
<tr><td></td><td>• Start: <code>state = 0</code>
• Input 2: <code>state</code> is 0. Input is not 1. → <code>state</code> remains 0.
• Input 1: <code>state</code> is 0. Input is 1. → <code>state</code> becomes 1.
• Input 3: <code>state</code> is 1. Input is not 1 or 2. → <code>state</code> remains 1.
• Input 1: <code>state</code> is 1. Input is 1. → <code>state</code> becomes 2.
• Input 1: <code>state</code> is 2. Input is not 3. → <code>state</code> becomes 1 (via <code>else</code>).
• Input 2: <code>state</code> is 1. Input is 2. → <code>state</code> becomes 0.
• Input 1: <code>state</code> is 0. Input is 1. → <code>state</code> becomes 1.
• Input 1: <code>state</code> is 1. Input is 1. → <code>state</code> becomes 2.
• Input 3: <code>state</code> is 2. Input is 3. → <code>state</code> becomes 0.
• Input 2: <code>state</code> is 0. Input is not 1. → <code>state</code> remains 0.
So, the output is 0.</td></tr>
<tr><td>D</td><td><code>a[i]=10-a[i]</code> // <code>a[i]=a[i]*9%10</code></td></tr>
<tr><td></td><td>Note that $3 + \mathbf{7} = 8 + \mathbf{2} = 9 + \mathbf{1} = 7 + \mathbf{3} = 1 + \mathbf{9} = 2 + \mathbf{8} = 10$, where the numbers in bold are the required outputs, and the numbers not in bold are as the array <code>a</code>.</td></tr>
<tr><td>E</td><td><code>b-a-(a_m>b_m)*40</code></td></tr>
<tr><td></td><td>We can obtain the resulting time from <code>b-a</code> if <code>a_m <= b_m</code>.
However, the <code>a_m > b_m</code> case will result in an extra 40 added to the answer. Hence we add the clause <code>-(a_m>b_m)*40</code> to get rid of it when <code>a_m > b_m</code>.</td></tr>
<tr><td>F</td><td><code>-i</code> // <code>-i&3</code> // <code>-i%4</code></td></tr>
<tr><td></td><td>We want to output the characters <code>s[0]</code>, <code>s[-1]</code>, <code>s[-2]</code>, <code>s[-3]</code> in order.</td></tr>
</table>

Section C

<table>
<tr><th>Q</th><th colspan="2">Answer and Explanation</th></tr>
<tr><td rowspan="2">G</td><td colspan="2"><code>X[i-1]+X[i]</code></td></tr>
<tr><td colspan="2">Partial sum can be used. After the loop on <code>i</code>, <code>X[i]</code> counts the number of balls with label not exceeding <code>i</code>.</td></tr>
<tr><td rowspan="2">H</td><td colspan="2"><code>A[i]*B[i]-A[i-1]*B[i-1]</code></td></tr>
<tr><td colspan="2">Note that <code>A[i]*B[i]</code> is the number of ways of getting a prize not exceeding <code>i</code>. Therefore, <code>A[i]*B[i] - A[i-1]*B[i-1]</code> gives the number of ways of exactly getting the prize <code>i</code>.</td></tr>
<tr><td rowspan="2">I</td><td colspan="2"><code>a==b and c==d</code> // <code>(a==b)&(c==d)</code></td></tr>
<tr><td colspan="2">The helper <code>condition(a,b,c,d)</code> should return true exactly when both equalities hold, i.e. $a = b$ and $c = d$. Hence, you should fill in <code>a==b and c==d</code>.</td></tr>
<tr><td rowspan="3">J</td><td>J1</td><td><code>x&y</code></td></tr>
<tr><td>J2</td><td><code>~x&z</code> // <code>z>x</code></td></tr>
<tr><td colspan="2">Note that Condition 3 is not necessary. Why is that so? That is because if $y = 1$ and $z = 1$, then:
• Either $x = 0, y = 1, z = 1$ (Condition 2 holds)
• Or $x = 1, y = 1, z = 1$ (Condition 1 holds)
Thus it is sufficient to combine the first two conditions by OR.</td></tr>
</table>

Paper 2 (C++)

Section A

Q	A	Explanation
1	D	C++ uses **zero-based indexing**, meaning the first element is at index 0. • `a[1]` refers to the second element: 4 • `a[4]` refers to the fifth element: 10
2	D	Each loop gets the least significant hex digit `y = x % 16`. To print digits in the correct order, we must place each new digit in front of the previous ones, so `push_front` is used.
3	D	Note that both break statements will not run. The first if condition `++i == j` is false as the pre-increment operator runs before the comparison operator. Since the first if statement increments `i`, the second if condition `i++ == j` will also be false. Tracing the program, each iteration increments `i` by 3, so the values of `i` after each iteration of the loop are 4, 7 then 10 eventually.
4	B	In C++, `','` is the comma operator. It evaluates its first and second operand sequentially. As the assignment operator `=` has a higher operator precedence than `,` in C++, `a, b = b, a;` is equivalent to `a; b = b; a;`, both the values of `a` and `b` are unchanged.
5	B	The first parameter is being passed by reference, so inside the function `f()`, the parameter `a` is referring to the variable `c` in `main()`. The second parameter is being passed by value, so it gets a copy of `a` without modifying its value in `main()`. The parameter `c` in `f()` refers to the variable `a` in `main()`. Initially in `f()`, `a` is 1, `b` is 2, and `c` is 2. Tracing the program, parameter `b` becomes $1 \times 2 = 2$, then `a` in `main()` becomes $2 - 1 = 1$, then `c` in `main()` becomes $1 + 2 = 3$. The value of `b` in `main()` remains unchanged at 3.

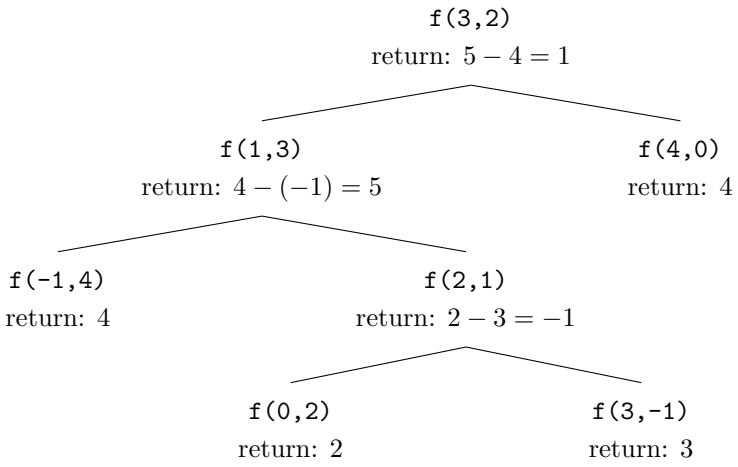
Section B

Q	Answer and Explanation
K	`4`

Here are the values of `i % 4` and `(8 - i) % 4` for different `i`:

`i`	`i % 4`	`(8 - i) % 4`
0	0	0
1	1	3
2	2	2
3	3	1
4	0	0
5	1	3
6	2	2
7	3	1
8	0	0
9	1	-1

So the answer is incremented by 2 for two times, once at $i = 1$ and once at $i = 5$. Therefore, the answer is 4.

Q	Answer and Explanation
L	`1`
	The recursion tree, as well as the return value for each of the calls, are shown below.  So, the output is 1.
M	`40`
	Note that the inner for-loop subtracts 1 from a prefix of `a` until `0` is met, and add 1 to `ans` every time a number is subtracted. Since index `i` is subtracted exactly `min(a[0], a[1], ..., a[i])` times, the answer is $13+9+5+5+5+3+0=40$.
N	`s[i]=='i'`
	It suffices to change all occurrences of `'i'`-s into `'j'`-s in the string `s`. Noting that `'i'` $+1=$ `'j'`, we can simply add 1 to every `s[i]` for which `s[i]` equals `'i'`.
O	`!(x&4) // x/4!=1 // x%8<=3 // (~x)&4 // x%8<4`
	We should return `false` if and only if $4 \le x \le 7$. The easiest way is to check the value of $\lfloor \frac{x}{4} \rfloor$.
P	`b[n]-2*b[u]+(2*u-n)*x`
	Let $u = f(a, x)$. Then $a[0], \ldots, a[u-1] < x$ and $a[u], \ldots, a[n-1] \ge x$. With prefix sums $b[i] = \sum_{k=0}^{i-1} a[k]$: $$\sum_{i=0}^{u-1} \lvert x - a[i] \rvert = \sum_{i=0}^{u-1} (x - a[i]) = ux - b[u],$$ $$\sum_{i=u}^{n-1} \lvert x - a[i] \rvert = \sum_{i=u}^{n-1} (a[i] - x) = (b[n] - b[u]) - (n-u)x.$$ Therefore, $$\sum_{i=0}^{n-1} \lvert x - a[i] \rvert = ux - b[u] + (b[n] - b[u]) - (n-u)x = b[n] - 2b[u] + (2u-n)x$$ And we obtain our expression.

Section C

<table>
<tr><th>Q</th><th colspan="2">Answer and Explanation</th></tr>
<tr><td>Q</td><td colspan="2">31</td></tr>
<tr><td></td><td colspan="2">When <code>f(n, k)</code> is called, the first k elements in the sequence a are initialized to 1 (i.e. $a_0 = a_1 = \ ... \ = a_{k-1} = 1$). Then, for all $k \leq i \leq n$, a_i is given by the sum of the previous k elements in the sequence (i.e. $a_i = a_{i-k} + a_{i-k+1} + ... + a_{i-1}$). Finally, the value of the n-th element (0-based) in the sequence is returned.
When $k = 3$, the sequence a is $1, 1, 1, 3, 5, 9, 17, 31, ...$ The 7-th element (0-based) in this sequence is 31. So, <code>f(7, 3)</code> returns 31.</td></tr>
<tr><td>R</td><td>R1</td><td><code>k</code></td></tr>
<tr><td></td><td>R2</td><td><code>2*a[i-1]-a[i-k-1]</code></td></tr>
<tr><td></td><td colspan="2">When <code>g(n, k)</code> is called, the first k elements in the sequence a are initialized to 1. By the definition of <code>f(n, k)</code>, $a_k = \sum_{j=0}^{k-1} a_j = \sum_{j=0}^{k-1} 1 = k$. So, <code>a[k]</code> should be initialized to <code>k</code>.
For all $k < i \leq n$, $a_i = \sum_{j=i-k}^{i-1} a_j = a_{i-1} - a_{i-k-1} + \sum_{j=i-k-1}^{i-2} a_j = a_{i-1} - a_{i-k-1} + a_{i-1} = 2a_{i-1} - a_{i-k-1}$.
Hence, <code>a[i]</code> should be defined using the above expression.</td></tr>
<tr><td>S</td><td colspan="2">1123000000</td></tr>
<tr><td></td><td colspan="2">We are given <code>p[n]</code> $= 10^n$ for $0 \leq n \leq 9$, and <code>r(x, 1)</code> rounding x to the nearest multiple of 10. Therefore:
• <code>r(1122444669, 1) = 1122444670</code>
• <code>r(1122444669, 2) = r(112244467 , 1) * p[1] = 1122444700</code>
• <code>r(1122444669, 3) = r(11224447 , 1) * p[2] = 1122445000</code>
• <code>r(1122444669, 4) = r(1122445 , 1) * p[3] = 1122450000</code>
• <code>r(1122444669, 5) = r(112245 , 1) * p[4] = 1122500000</code>
• <code>r(1122444669, 6) = r(11225 , 1) * p[5] = 1123000000</code>
P.S. Building upon this, the values for which <code>r(x, 6)</code> $= 1123000000$ are $x \in [1122444445, 1123444444]$.</td></tr>
<tr><td>T</td><td colspan="2"><code>x-p[n]/18</code> // <code>x-x%p[n]/9</code></td></tr>
<tr><td></td><td colspan="2">From the above result we can conclude that <code>r(x, n)</code> performs rounding on each of the last n digits of x in a sequential manner. For each output A being a multiple of 10^n,
• observe that the values for which <code>r(x, n)</code> $= A$ are $x \in \left[A - 10^n \times \frac{5}{9}, A + 10^n \times \frac{4}{9}\right)$; and
• our goal is to complete the function <code>rnd</code> such that the values for which <code>rnd(x, n)</code> $= A$ should be $x \in \left[A - 10^n \times \frac{1}{2}, A + 10^n \times \frac{1}{2}\right)$.
Hence the expression to be filled in should be able to map x from the interval $\left[A - 10^n \times \frac{1}{2}, A + 10^n \times \frac{1}{2}\right)$ to $\left[A - 10^n \times \frac{5}{9}, A + 10^n \times \frac{4}{9}\right)$. Two ways to do this are:
• <code>x-p[n]/18</code>: maps through linear shift, or
• <code>x-x%p[n]/9</code>:
– maps $\left[A - 10^n \times \frac{1}{2}, A\right)$ to $\left[A - 10^n \times \frac{5}{9}, A - 10^n \times \frac{1}{9}\right)$, and
– maps $\left[A, A + 10^n \times \frac{1}{2}\right)$ to $\left[A, A + 10^n \times \frac{4}{9}\right)$
Note that the expression should not involve <code>p[n-1]</code>, otherwise out-of-bound errors occur when $n = 0$.</td></tr>
</table>

Appendix: Marks Conversion Table for Paper 2 Python Version

There were two papers. Candidates were required to answer ALL questions in Paper 1. In Paper 2, candidates could choose EITHER the Python or C++ version. In the marking process, the marks for the Python version were converted to the marks on the scale for the C++ version using the tables below. For example, a score of 10 marks scored by a candidate taking the Python version would be converted to 8 marks on the C++ version scale.

These tables were generated by an equating method based on the performance of the contestants in this year's competition.

Python Version	C++ Version
0	0
0.5	0.5
1	0.5
1.5	1
2	1
2.5	1.5
3	1.5
3.5	2
4	2.5
4.5	2.5
5	3
5.5	3.5
6	3.5
6.5	4
7	5
7.5	5.5
8	6
8.5	6
9	7
9.5	7.5
10	8

Python Version	C++ Version
10.5	8.5
11	9
11.5	9.5
12	10
12.5	10.5
13	11
13.5	11.5
14	11.5
14.5	12
15	12.5
15.5	13
16	13.5
16.5	14
17	14.5
17.5	14.5
18	15
18.5	15.5
19	16
19.5	16.5
20	20