Hong Kong Olympiad in Informatics 2024/25
Heat Event (Senior Group)
Official Solution

Statistics (N = 458)
Full mark = 45. Maximum = 44. Median = 17. Advance to Final = 26 marks or above.

Paper 1 (Compulsory Part)

Section A

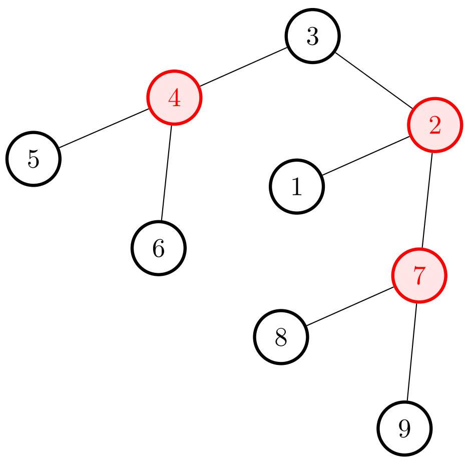
Q	A	Explanation
1	D	1 MB = 1024 KB = 1024 × 1024 bytes, so 400 MB = 400 × 1024 × 1024 bytes.
2	A	For each `i` from 0 to 4, calculate the index `(i*3)%5` and update `a[i]`:

i	(i*3)%5	Instruction	a
0	(0*3)%5=0	`a[0] = a[0]`	`[1, 2, 3, 4, 5]`
1	(1*3)%5=3	`a[1] = a[3]`	`[1, 4, 3, 4, 5]`
2	(2*3)%5=1	`a[2] = a[1]`	`[1, 4, 4, 4, 5]`
3	(3*3)%5=4	`a[3] = a[4]`	`[1, 4, 4, 5, 5]`
4	(4*3)%5=2	`a[4] = a[2]`	`[1, 4, 4, 5, 4]`

		The final output is `1 4 4 5 4`.

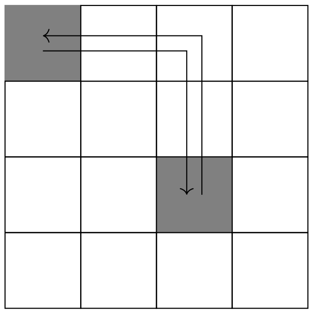
3	C	Simplify both expressions using De Morgan's Theorem:
		i. `NOT(((NOT A) OR B) AND (A OR (NOT B)))`
		`= NOT((NOT A) OR B) OR NOT (A OR (NOT B))`
		`= (A AND (NOT B)) OR ((NOT A) AND B)`
		`= A XOR B`
		ii. `NOT((A AND B) OR ((NOT A) AND (NOT B)))`
		`= NOT(A AND B) AND NOT((NOT A) AND (NOT B))`
		`= ((NOT A) OR (NOT B)) AND (A OR B)`
		`= A XOR B`

4	B	Statmenet (i) is true, as merge sort involves merging two sorted subarrays into a sorted array. Statement (ii) is false because quicksort selects an arbitrary pivot element from the current (sub)array; the pivot may not necessarily be the smallest element. Statement (iii) is true. It is a description of the insertion sort algorithm.

Q	A	Explanation
5	B	 The chosen vertices are colored red. For all leaf vertices such as vertex 1, we must choose either themselves or their only neighbor in order to cover the edge in between. We always want to choose the neighbor as it has the possibility of covering more edges. Therefore, we will choose vertices 2, 4, and 7. Choosing them is sufficient to cover all edges.
6	A	i. This string can be reduced to an empty string as $\texttt{AA}\underline{\texttt{AAB}}\texttt{BBAA}\rightarrow\texttt{A}\underline{\texttt{AB}}\texttt{B}\underline{\texttt{A}}\texttt{A}\rightarrow\underline{\texttt{ABA}}$. ii. Each operation will remove one B and at least one A on its left. In the first seven characters of string ii, there are four B and three A. It's impossible to remove at least one A on each B's left. Hence, it's impossible to reduce it to an empty string. iii. Each operation will remove two A and one B, so the ratio of the number of A and the number of B should be 2:1. Hence, it's impossible to reduce it to an empty string.
7	C	There are two cases: Case 1: If the number of crewmates is odd, player 5 and player 4 is a crewmate and an impostor respectively. Because of this, player 3 tells the truth and is a crewmate. Therefore, player 2 is an impostor, which implies that player 1 is a crewmate. The number of crewmates is 3, which is odd. Case 2: If the number of crewmates is even, player 5 and player 4 is an impostor and a crewmate respectively. Because of this, player 3 lies and is a impostor. Therefore, player 2 is a crewmate, which implies that player 1 is a impostor. The number of crewmates is 2, which is even. So the minimum number of crewmates is 2.
8	C	Since the value popped from the last operation is 7, it must be pushed from B to A during operation 5 and must be popped immediately after. For B, the element that was first pushed has to pop first, so B is a queue. For A, the element that was last pushed has to pop first, so A is a stack.
9	B	The final round being the first seed versus the second seed requires that the two seeds are from different halves of the tournament bracket at the beginning. When the first seed is in one half, there are 64 remaining candidates in the other half for the second seed out of $128 - 1 = 127$ total candidates. The probability that the second seed is in a different half than the first seed is $\frac{64}{127}$, which is closest to 0.5.

Q	A	Explanation
10	C	Denote cell (i, j) to be the i^{th} cell counted from the top and j^{th} cell counted from the left. Observe that if we starts at top-left corner cell $(1, 1)$, then move to any other cell (r, c), and go back to $(1, 1)$ using the original route, we visit every cell in the middle twice and we only toggled the colours of cell $(1, 1)$ and (r, c).

(1,1)	(1,2)	(1,3)	(1,4)
(2,1)	(2,2)	(2,3)	(2,4)
(3,1)	(3,2)	(3,3)	(3,4)
(4,1)	(4,2)	(4,3)	(4,4)



Therefore, if we repeat the routine for all black cells other than cell $(1, 1)$, we can toggle them all white, and cell $(1, 1)$ would be toggled (number of black cells other than cell $(1, 1)$) times. We can infer from this that whenever the total number of black cells is even, we can turn all cells white, which is the case for both (i) and (ii).
Alternatively, if the number of black cells is odd, it is impossible to turn all cells white, as every move we change the parity of the number of black cells. Since we must start and end at cell $(1, 1)$, we must have change the parity even number of times, and so there would be odd number, i.e., non-zero number of black cells remains in the board.

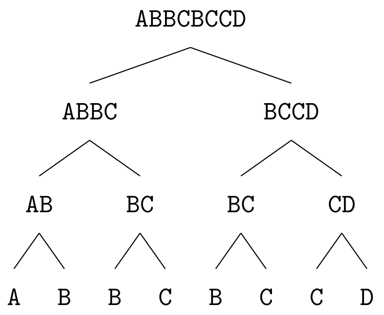
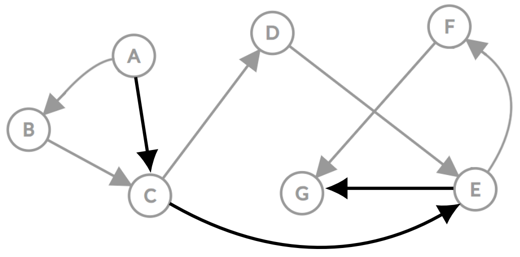
Section B

Q	Answer and Explanation
A	`2,1,2`
	Initially, we have $a = 3$, $b = 2$, $c = 1$. After running the second line, $a = 3 - 2 + 1 = 2$, while b and c are left unchanged. After running the third line, $b = 2 - 2 + 1 = 1$, while a and c are left unchanged. After running the fourth line, $c = 2 - 1 + 1 = 2$, while a and b are left unchanged. Hence, the final output is `2,1,2`.
B	5
	Since 24 is a multiple of 6, we can distribute the candies originally given to the 24 people among 6 people evenly. This allows us to simplify the problem to finding the number of remaining candies after distributing 17 candies among 6 people. Thus, the final answer is $17 \mod 6 = 5$.
C	b, f
	The original graph is not strongly connected because nodes 0 and 3 have no incoming edges, making them unreachable from other nodes. Strong connectivity can be achieve by building the cycle `0 → 1 → 4 → 3 → 2 → 0`. By reversing edges **b** and **f**, we create a cycle that contains all vertices, thus making the entire graph strongly connected. We can simply check that no single reversal solution exists. So at least two reversals were necessary to provide incoming paths for nodes 0 and 3, and this solution is minimal. Thus, the set of edges to be reversed is **b, f**.

Q	Answer and Explanation
D	`aaabbabb // aabaabbb // aaaababb // aababbbb`
	Each `'a'` increases the counter `a` by 1 and each `'b'` increases the counter `b` by the value of `a` at that moment. Therefore, the string corresponds to the sum of some increasing integers. The total number of `'b'`s is the count of numbers in the sum, and the total number of `'a'`s is at least the value of the last integer in the sum. Also, the total number of `'a'`s and `'b'`s cannot exceed 8. From these, we can express 14 as: $3+3+4+4$, $2+4+4+4$, $4+5+5$, or $2+3+3+3+3$, which correspond to `aaabbabb`, `aabaabbb`, `aaaababb`, and `aababbbb` respectively.
E	540
	Each candy must be assigned to one of 3 students, hence we have 3^6 assignments. However, we have to subtract the number of assignments that result in one or more students having no candies. For each student, there are 2^6 assignments that result in said student having no candies. There are also 3 assignments where 2 students have no candies are double-counted, so we should add them back to the answer. Therefore, there are a total of $3^6 - 3 \times 2^6 + 3 = 540$ assignments.
F	`4`
	In the final answer, each element in `a` is taken 7 times and each element in `b` is taken 4 times. Since `x ^ x = 0`, the answer is exactly the XOR sum of `a`, as XORing something twice cancels each other out. Therefore, the answer is `4 ^ 5 ^ 7 ^ 2 = 4`.

Section C

Q	Answer and Explanation
G	`B`
	Note that every time a copy of S is pasted to the end of the sequence, the length of the sequence will be multiplied by 2. Hence, the length of S after every 'copy' action is always a power of 2. Note that 64 is a power of 2, hence we know that the 65-th character of S is the first character of a copy. We know that the first character of S is `A`, hence the first character of a copy is `B`.

Q	Answer and Explanation
H	G
	First, let’s take a look at the recursion tree of the first 8 characters of S.  ABBCBCCD ABBC BCCD AB BC BC CD A B B C B C C D Observe that in this recursion tree, traversing down to the right corresponds to performing a ‘copy’ action and shifting the letters by one position in the alphabet. Hence, we would like to find the number of ‘right turns’ required to reach the 160-th character. To achieve this, we first convert the position to a 0-based index, then to its binary representation. Observe that after such conversion, a `0` digit in the binary number represents a ‘left turn’, while a `1` digit represents a ‘right turn’. Hence, the problem reduces to counting the number of `1`s in a binary number. To find the answer, we first convert the 160-th position to its 0-based index, 159, then convert 159 to its binary representation, resulting in `10011111`. Since there are 6 `1`s in this binary number, we have made 6 alphabet shifts to `A`. Thus, the answer is the 7-th alphabet, `G`.
I	8
	From A, there is 1 way to get to B (1 directly from A), 2 ways to get to C (1 directly from A and 1 from B), 3 ways to get to D (1 from B and 2 from C), 5 ways to get to E (2 from C and 3 from D), and 8 ways to get to F (3 from D and 5 from E).
J	
	After adding the arrows as shown in the above image, there will be 2 ways to go from A to C, 2 ways to go from C to E, and 2 ways to go from E to G. Thus, there will be in total of $2 \times 2 \times 2 = 8$ ways to go from A to G as each of the 3 choices are independent.

Paper 2 (Python)

Section A

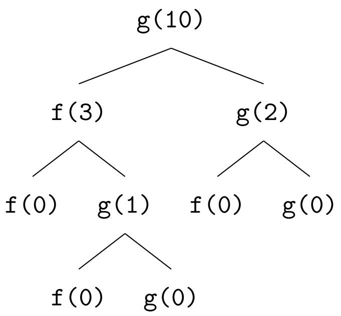
Q	A	Explanation
1	D	The variable `x` is initialized to `4`. The if condition `2 ** x == 16` evaluates to true, causing `x` to be updated to `5`. The program then prints the final value of `x` twice, resulting in the output `5 5`. In Python, control-flow blocks like if statements do not create a new scope. Therefore, the assignment `x = 5` inside the if block modifies the original variable `x` in the surrounding scope.
2	C	Let's compare the amount of work the computer has to do for each operation on the list of 100000 numbers: ii. `L.pop(-1)`: This is the fastest operation. To remove the last number from the list, the computer can simply access it directly and remove it. No other numbers in the list need to be moved in the computer's memory. i. `L.pop(0)`: This operation is slower. When you remove the very first number, it leaves a gap at the beginning of the list. To fill this gap, the computer must shift every single one of the remaining 99999 numbers one position forward. iii. `L.sort()`: This is the slowest operation. Python uses a hybrid sorting algorithm similar to Merge Sort, which divides the array and recursively sort them. Each element can be processed multiple times, making it much more work than just removing one element. Therefore, the order would be (ii), (i), (iii). As a side note, the time complexity of operations (i), (ii), and (iii) in Big-O notation are $O(n)$, $O(1)$ and $O(n \log n)$ respectively.
3	D	`'+'.join(a)` is `'H+K+O+I'`. The separator is used only when multiple items are passed in `print()`.
4	*	This question is testing for `ZeroDivisionError` in Python, namely, when you divide an integer by zero, the program will raise a runtime error and crash. We can split the input `a` into four cases: • If `a >= 2`, then `b = 1 // a = 0`, and `1 // b` causes the program to crash. • If `a = 1`, then `b = 1 // a = 1`, and `c = 1 // b = 1`. The program does not crash. • If `a = 0`, then `1 // a` causes the program to crash. • If `a < 0`, since Python integer division always returns the floored value after division, `b = 1 // a = -1` and `c = 1 // b = -1`. The program does not crash. Therefore, the total number of input values `a` such that the program does not crash is $2^{31} + 1$. **Remark:** Unfortunately, during the actual contest, the case `a < -1` is mistakenly thought to lead to a program crash, due to different integer division behaviors in Python and C++. Therefore, the answer that is marked as correct is `a = -1 or a = 1`, which leads to the answer of 2. We are sorry for the mistake and would like to apologize to the participants that are affected by this.

Q	A	Explanation
5	C	The program builds the output string by processing the input integer in two-digit chunks, from right to left. To reconstruct the input number that yields the output `HkOi`, we must work backward from the target string. We find the index for each character in the `alphabets` string and use these indices to form the required integer. • The first character of the output is `H`, which corresponds to index **7**. This means the last two digits of the input integer must be **07**. • The second character is `k`. The uppercase letters occupy indices 0 - 25, so the lowercase `'k'` (11th letter, index 10) is at index $26 + 10 = $ **36**. These are the next two digits. • The third character is `O`, which is the 15th letter of the alphabet, corresponding to index **14**. • The fourth character is `i`. The lowercase `'i'` (9th letter, index 8) is at index $26 + 8 = $ **34**. To form the original integer, we assemble the required two-digit numbers in reverse order of how they were generated: `34`, then `14`, then `36`, and finally `07`. So the answer is **34143607**.

Section B

Q	Answer and Explanation
A	`14`
	First of all, `magic()` doesn't swap the two values but instead assigns `a[2]` to `a[0]` only. Actually, the statement `a[2] = a[0]` is useless. In the first step, since we pass an array to `magic()`, all changes to the array inside `magic()` affects the actual array itself, even outside of the function. Next, when we assign `b[2]` to `b[0]`, what actually happens is that `b[2]` and `b[0]` become references to the same underlying list. All changes in `b[2]` will be reflected in `b[0]`, and vice versa. Therefore, during the `magic(b[0])` call, when we assign `b[0][2]` (9) to `b[0][0]`, the value of `b[2][0]` will become 9 as well, and similarly for assigning 0 to `b[0][2]`. After executing the assignment operations, the final array would be `[[9, 8, 0], [4, 5, 6], [9, 8, 0]]`, so the answer is $9 + 5 + 0 = 14$.
B	`4`
	By tracing the program's execution, we can observe the sequence of values that the variable `x` takes after each iteration of the loop: 1, 9, 2, 8, 4, 9, 2, 8, 4, ... After the first iteration, the sequence enters a repeating cycle of four values: **(9, 2, 8, 4)**. So after 2025-th iteration, `x` will be the last element in the cycle, which is 4.

Q	Answer and Explanation
C	`13`

	The recursion tree is as follows.  g(10)
	f(3)
	g(2)
	f(0)
	g(1)
	f(0)
	g(0)
	f(0)
	g(0) Therefore, the answer is $3+3+2+3+2=13$.

D	`3<=i<=5 // i//3==1`
	We want to add `i` to `total` when `i` is between 3 and 5, inclusive. One may achieve this with either chaining comparisons, or by divison.
E	`a[:idx]+a[idx+1:]`
	This is an implementation of selection sort. For every iteration of `f(a)`, `a[idx]` is the minimum element in the passed list `a`. `a[idx]` has to be removed before being passed to the next iteration of `f(a)`, such that the new list `a` excluding `a[idx]` has a new minimum element to be recursively sorted.
F	`1-x,1-y // 1-y,1-x`
	The function generates a random coordinate `(x, y)` within the unit square defined by $0 \leq x, y \leq 1$. This square is bisected by the line $x+y=1$ into two triangles of equal area. The objective is to return a point from the lower triangle, where the condition $0 \leq x+y \leq 1$ holds. If the randomly generated point already falls within this lower triangle, the function returns it directly. However, if the point lies in the upper triangle (where $x+y>1$), it must be transformed. The expression `(1 - x, 1 - y)` maps any point from the upper triangle to a corresponding point in the lower one. This transformation is equivalent to a 180-degree rotation around the center of the square, which preserves the uniform distribution of the random points. Therefore, the answer is `1 - x, 1 - y` or `1 - y, 1 - x`.

Section C

Q	Answer and Explanation

G	`m==4 or m==6 or m==9 or m==11 //`
	`m in (4,6,9,11)`

	One may observe that there are only 4 different months with 30 days, and the space given is quite abundant. It suffices to check whether `m` falls into one of those 4 values.

H	`(m>7)^(m%2) //`
	`(m<8)==(m%2)`

	We may observe that the odd months have 31 days until July, after which the even months have 31 days instead. We may use the exclusive-or operator `^` or the equality operators `==`, `!=` to combine the conditions “`m` is odd” and “`m` is at most 7” (or their negations).

Q	**Answer and Explanation**	
I	I1	`i*2`
	I2	`i*2+1`
	In the `i`-th iteration, the program takes one element from the first half and one element from the second half. These two elements should be assigned back to `a` in order. Since each iteration involves 2 elements, their indices will be `i*2` and `i*2+1` respectively.	
J	J1	`n//4`
	J2	`l+n//4+i`
	J3	`l+n//2+i`
	This program uses the divide and conquer technique to transform `a` into its interleave list. It first performs a transformation on `a`, then recursively solves two sub-problems, namely, finding the interleave list for the left half and the right half.	
	It would be useful to observe which elements will be in the left half of the final list. We notice that the left half of the final list contains the elements from the first quarter (`a[0]` to `a[n//4-1]`) as well as the elements from the third quarter (`a[n//2]` to `a[n//2+n//4-1]`). Therefore, the right half of the final list contains the elements from the second quarter (`a[n//4]` to `a[n//2-1]`) as well as the elements from the fourth quarter (`a[n//2+n//4]` to `a[n-1]`). This suggest us to swap the elements in the second quarter with that in the third quarter.	
	The final step is to check that this algorithm is correct. It suffices to check that the interleave of an list is equal to the concatenation of the following two parts: (1) the interleave list of the first and third quarter, and (2) the interleave list of the second and fourth quarter.	
	Extra: Although this algorithm works in $O(n \log n)$ time, it does not consume any extra memory!	

Paper 2 (C++)

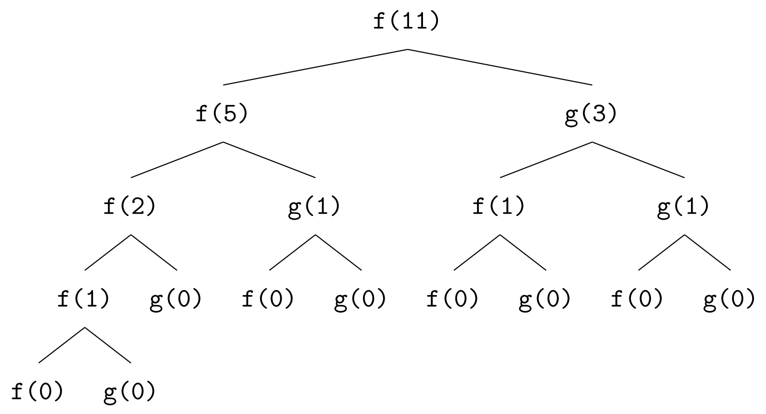
Section A

Q	A	Explanation
1	C	In C++, `^` is the bitwise XOR operator, `**` is not a valid arithmetic operator, and `<<` is the left shift operator. `1 << x` shift the bits of 1 by `x` positions to the left, calculating 2 to the power of `x`.
2	B	The loop runs for $i = 1$ and $i = 2$. Thus, `sum` $= 0 + a[1] + a[2] = 0 + 2 + 7 = 9$. So the output is `9`.
3	C	Let's compare the amount of work the computer has to do for each operation on the `vector` of 100000 integers: iii. `a.pop_back()`: This is the fastest operation. To remove the last integer from the vector, the computer simply decrements the vector's internal size counter. No other integers in the vector need to be moved in the computer's memory. This makes it an extremely quick, direct process. i. `reverse(a.begin(), a.end())`: This operation is slower. In order to reverse the vector, nearly every integer must be moved to a new position in memory. The first element must end up in the last position, the second element in the second-to-last position, and so on. This requires passing through the vector and moving almost every single element once. ii. `sort(a.begin(), a.end())`: This is the slowest operation. C++ uses a hybrid sorting algorithm involving Quicksort, which divides the vector and recursively sort them. Each element can be processed multiple times, making it much more work than just reversing the vector. Therefore, the order from fastest to slowest is (iii), (i), (ii). As a side note, the time complexity of operations (i), (ii), and (iii) in Big-O notation are $O(n)$, $O(n \log n)$, and $O(1)$ respectively.
4	A	The `else` part is executed as `x + y != 2024`. As the new variable `x` is defined within a scope, it is only accessible within that scope. Hence `x` in the `cout` statement will reference the variable created on line 2 of the program, and has value 20.
5	D	In C++, the parameters are evaluated first before calling the function. Therefore, `g()` is first called before `f(2)`, so the answer is `32`.

Section B

Q	Answer and Explanation
K	`3,19,12`
	When `foo(5, 7)` is passed: On Line 1, global variable `c` sums the two parameters passed in `foo`. Therefore global variable `c` $= 5 + 7 = 12$. On Line 2, parameter `a` references global variable `b` and sums parameter `b` and global variable `c`. Therefore global variable `b` $= 7 + 12 = 19$. On Line 3, note that parameter `b` is updated rather than a global variable which does not change the final answer.
L	`0`
	Observe that `(x & 2) >> 1` accesses the second least significant bit of `x`, `x & 1` accesses the least significant bit of `x`, and `x % 4` outputs the value represented by the least two significant bits of `x`. We can consider the least two significant bits of `x` and the elements of `a`. The least two significant bits of `x` follow a cycle of `00, 10, 01, 11, 00, 10, 01, 11, 00, 10, ...`, and the 2024-th element in the cycle (0-based) is `00` which represents 0.

Q	Answer and Explanation
M	`13`

	The recursion tree is as follows.  f(11)
	f(5)
	g(3)
	f(2)
	g(1)
	f(1)
	g(1)
	f(1)
	g(0)
	f(0)
	g(0)
	f(0)
	g(0)
	f(0)
	g(0)
	f(0)
	g(0) Therefore, the answer is $2+1+1+2+1+2+1+2+1 = 13$.

N	`i%2 // i&1`
	The output increases only when n is odd, and matches the sum of odd numbers up to n. That means we only add i when i is odd. So the answer is `i%2`.

O	`(x-'a'+3)%26+'a' //`
	`x+3>'z'?x-23:x+3`

	Note that all non-space characters in `str1` has to be shifted cyclically by 3 letters (e.g. `h` to `k` and `y` to `b`). Update the ASCII values of letters and handle cyclic shifts by taking `mod 26` or case-handling.
P	`++b[a[i]]`
	The program implements the counting sort algorithm. We count occurrences of each value in `a` into the frequency array `b`. The final loops then print values from 10 down to 0, each repeated `b[i]` times, yielding `10 9 8 7 6 5 4 3 3 2 2 2 1 0`.

Section C

Q	Answer and Explanation	
Q	Q1	`t/60`
	Q2	`t%60/10`
	Q3	`t%10`
	The hours are computed as `t/60` and it will always be a single digit since $t < 600$. The minutes are computed as `t%60`, ranging from 0 to 59. The tens digit is `t%60/10`, and the units digit is `t%60%10`, or `t%10` when simplified.	
R	`7`	
	`f(x)` returns the number of non-empty subsets of the ones in the binary representation of `x`. As 26 in binary is 11010, and there are 3 ones, the number of non-empty subsets is $2^3 - 1 = 7$. Alternatively, all values of `i` can be obtained by dry-running the program.	
S	`i&x; i // i|x; x // x^i; x-i` (interchangable)	
	In the loop, the program should check if ones in the binary representation of `i` is a subset of ones in the binary representation of `x`.	
T	T1	`(1<<i)&x // (x>>i)&1`
	T2	`(1<<cnt)-1`
	`cnt` is the number of ones in the binary representation of x, and the answer should be $2^{\texttt{cnt}} - 1$.	

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 11.5 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.

Raw Marks	Equated Marks
0	0
0.5	0.5
1	1
1.5	2
2	2.5
2.5	3
3	3.5
3.5	4
4	4.5
4.5	6
5	6.5
5.5	7.5
6	8
6.5	8.5
7	9
7.5	9.5
8	10
8.5	10.5
9	10.5
9.5	11
10	11.5

Raw Marks	Equated Marks
10.5	11.5
11	12
11.5	12.5
12	13
12.5	13.5
13	14
13.5	14.5
14	15
14.5	15.5
15	16
15.5	16.5
16	17.5
16.5	18
17	18.5
17.5	19
18	19.5
18.5	20
19	20
19.5	20
20	20