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

Statistics (N = 382)
Full mark = 45. Maximum = 42.5. Median = 17. Advance to Final = 25.5 marks or above.

Paper 1 (Compulsory Part)

Section A

Q	A	Explanation
1	D	The first card cannot be on the table, as if the side facing down shows "HKOI", the side facing up must show "Fun". The second and third cards can be on the table, as neither side of the cards shows "HKOI" and hence the rule does not apply.
2	A	In both languages, `%` is the modulo operator `^` and is the binary XOR operator (not the exponential operator normally seen in mathematical context). The binary XOR of 10 and 2 is 8, and the remainder of any positive integer divided by 8 must be smaller than 10, so the final floor division must result in 0.
3	D	`c` iterates through "HKOI" one by one and is appended to `cur`, so `cur` will be "H", "HK", "HKO", and "HKOI". Each time `cur` changes, it is appended to `ans`, `ans` is thus the concatenation of "H", "HK", "HKO", and "HKOI".
4	B	Linear search works by checking each element one by one until the target is found or the entire array is examined. It does not require the array to be sorted (A is false). Binary search does not involve parallel linear search (C is false), and works on sorted array of any length, not just powers of 2 (D is false).
5	B	i. `((X AND Y) AND ((NOT X) AND (NOT Y)))` ≡ `((X AND (NOT X)) AND (Y AND (NOT Y)))` (by commutative law) ≡ `(FALSE AND FALSE)` (by inverse law) ≡ `FALSE` (by idempotent law) ii. `((X AND (NOT X)) OR (NOT Y))` ≡ `(FALSE OR (NOT Y))` (by inverse law) ≡ `NOT Y` (by null law) iii. `((X OR Y) OR (X OR (NOT Y)))` ≡ `((X OR X) OR (Y OR (NOT Y)))` (by commutative law) ≡ `(X OR (Y OR (NOT Y)))` (by idempotent law) ≡ `(X OR TRUE)` (by inverse law) ≡ `TRUE` (by null law)

Q	A	Explanation
6	C	i. False. A stack is last-in first-out (LIFO), so after pushing 3, 1, 4, 5 the top is 5 (not 3). ii. True. A queue is first-in first-out (FIFO), so the front remains the first inserted element, 3. iii. True. A singly linked list traversal starts at the head; visiting $3 \to 1 \to 4 \to 5$ means the head points to 3.
7	D	The program counts for every character in `s`, the number of characters in `s` (including itself) that are the same as it. Hence, for a character that occurs n times in `s`, it will contribute n^2 to the answer. There are 2 characters (`4`, `9`) that occurs once and 4 characters (`2`, `0`, `-`, `1`) that occurs twice. Hence the answer is $2 \times 1^2 + 4 \times 2^2 = 18$.
8	B	i. For $n = 5$, after Alice takes 1 stone, 4 stones remain in the pile. Bob has two choices: 1. Take 1 stone, leaving 3 stones in the pile. 2. Take 2 stones, leaving 2 stones in the pile. No matter what Bob chooses, Alice can then take 2 stones, leaving 1 or 0 stones in the pile. Since Bob cannot take 2 or 3 stones, he must lose the game. ii. For $n = 6$, after Alice takes 1 stone, 5 stones remain in the pile. If Bob takes 1 stone, Alice will be faced with the situation in (i), and she must lose the game. Therefore, Bob has a winning strategy.
9	A	The first two actions are the only ways to add the letters `a` and `b` to the string, and both add one `a` and one `b` to the string. Action 3 only removes the `S`, which can only be done once at last. That means, if we cut the final string in the middle, the left string must be "symmetric" to the right string. Here, "symmetric" means if the left part is `a`, then the right part must be `b`, and vice versa. (i) and (ii) meet this criteria while (iii) does not.
10	D	As `flip()` returns 0 or 1 independently, if there are multiple `flip()` calls in a statement, their value may or may not be the same. Therefore, `x` can be any integer between 0 and 13 inclusive, and none of (i), (ii) or (iii) must be true.

Section B

Q	Answer and Explanation
A	`2026`
	Since `a` is not equal to `b`, the value of `b` is assigned to `a`, which is 2025. Then, the value of `a + 1` is assigned to `b`, which is $2025 + 1 = 2026$.
B	`hkoi2025 // hkoi2024/26` Or any other reasonable answers
	We need `s` < `u` < `t` for the program to output the desired string (`"ioi2025"`). String comparison is done lexicographically, so a reasonable answer would be `"hkoi2025"` as the first 7 characters are the same as that of `s` and the 8th character is larger, and the first 8 characters are the same as that of `t` and it ends there whereas `t` has characters afterwards.

Q	Answer and Explanation
C	$\frac{5}{8}$ // 0.625 // 62.5%
	Given that the sum of the two numbers are even, the numbers must be either both odd or both even. There are five odd numbers from 1 to 9, so there are $C^5_2 = \frac{5\times4}{1\times2} = 10$ ways to choose two odd numbers. Similarly, there are $C^4_2 = \frac{4\times3}{1\times2} = 6$ ways to choose two even numbers. Therefore, the probability that both the two numbers are odd is $\frac{10}{10+6} = \frac{5}{8}$.
D	`fhi`
	Due to `f()` returning `false` and short-circuit evaluation in Python/C++, `g()` is not executed. Note that `&&` has a higher precedence than `||`.
E	186
	For Alice and Bob to arrive at the same point, we can deduce that $2024 + 5a = 4040 - 11b$ where a, b are the number of steps Alice and Bob takes respectively. Rearranging the terms, we have $5a + 11b = 4040 - 2024 = 2016$. Taking modulo 5 both sides, we have $b \equiv 1 \pmod 5$, and in turns we can substitute $b = 5k + 1$, where k is a non-negative integer. We get $a = \frac{2016-11(5k+1)}{5} = 401 - 11k$. Since $a + b = (401 - 11k) + (5k + 1) = 402 - 6k$, to minimize $a + b$, we should maximize the value of k. We know $a \geq 0 \implies 401 - 11k \geq 0 \implies k \leq 401/11 \approx 36.45$, so the maximum k is 36. Therefore, the maximum value of $a + b$ is $402 - 6 \cdot 36 = 186$.
F	19
	Each iteration increments `ans` and `i` by 1, and decrements `a` by 1, until `i` $\geq$ `a`. Therefore, the loop body runs for $\lceil \frac{\texttt{a}}{2} \rceil$ times. For `a` $= 18$, `ans` $= 9$ and "Cool" is printed, while for `a` $= 19$, `ans` $= 10$ and "Not cool" is printed. Hence, "Cool" is the output for `a` between 0 and 18 inclusive.

Section C

Q	Answer and Explanation
G	`j,j+1`
	`g(n)` is an implementation of bubble sort. A typical bubble sort consists of $n - 1$ iterations, where in each iteration all elements are compared with the one after it one by one. After the i-th iteration, the largest i elements are sorted in their correct position.
H	`i,j`
	`h(n)` is an implementation of selection sort. A typical selection sort consists of $n - 1$ iterations, where in the i-th iteration the minimum element among `a[i]` to `a[n]` is swapped to `a[i]`.
I	15
	Each of 1, 2, . . ., 9 appears exactly once so the total sum of numbers of the whole grid is 45. So the sum in each row (and column) is $45 \div 3 = 15$.

Q	Answer and Explanation
J	10

The pairs of numbers that share a row or column with 9 must be $\{2, 4\}$ and $\{1, 5\}$. Note that 9 cannot be located in a corner, as this would violate the condition of opposite corners to have the same sum Z. Also note that 9 cannot occupy the cell in the centre, because the four remaining numbers (namely 3, 6, 7 and 8) cannot be grouped into two pairs of equal sums, again violating the condition.

Up to rotations and reflections, the grid can thus be represented as

A_1	9	A_2
C_1	B_1	C_2
C_3	B_2	C_4

where the elements of sets A and B are $\{1, 5\}$ and $\{2, 4\}$ in some order, and the set C is $\{3, 6, 7, 8\}$.

Assume, for contradiction, that $\{A_1, A_2\} = \{1, 5\}$. Without loss of generality, let $A_1 = 1$ and $A_2 = 5$ (since otherwise one can simply reflect the grid). The only two elements in $\{3, 6, 7, 8\}$ with a common difference of $A_2 - A_1 = 4$ is 3 and 7. To satisfy the opposite corner condition, we must then place $C_3 = 3$ and $C_4 = 7$. However, no assignment of C_1 and C_2 from the remaining $\{4, 8\}$ satisfies the condition $Y = A_1 + C_1 + C_3 = A_2 + C_2 + C_4$. This leads to a contradiction.

Thus, we must have $\{A_1, A_2\} = \{2, 4\}$. Furthermore, the only two elements in $\{3, 6, 7, 8\}$ with difference 2 are 6 and 8. Therefore, the answer can only be $2 + 8 = 4 + 6 = 10$.

Remark: one may observe that the 3×3 magic square satisfies all the conditions, directly arriving at the answer to be $15 - 5 = 10$.

Paper 2 (Python)

Section A

Q	A	Explanation
1	C	(i) is false because `bool` can only be `True` or `False`, but not `None`. (ii) is false because `str` can be a sequence of character of length `0`. (iii) is true because `float` can be `nan`, stands for `Not A Number`, which is used to represent undefined or unrepresentable numerical results.
2	C	Mathematically we have $21 \div 5 = 4.2$ and $21 \div 5.6 = 3.75$. In Python, '`/`' performs classic division and always return a float, so `21 / 5` returns `4.2`; '`//`' performs floor division which discards the fractional part, therefore `21 // 5` returns `4` (an integer), and `21 // 5.6` returns `3.0` (integer result in float).
3	B	Underflow: The result of an arithmetic calculation is too small in magnitude, such that it cannot be accurately represented by the standard floating-point data type. Overflow: The result of an arithmetic calculation is too large or too small, such that it exceeds the limit of the data type. Precision error: Computer's inability to represent some decimals exactly. Truncation error: The difference between the actual value and the truncated value of a function. In this question, 1e+200 $\times$ 1e+200 = 1e+400, which exceeds the upper limit ($\approx$ 1e+308). So, the answer is overflow.
4	C	The function `range(20, 5, -3)` generates a sequence of five numbers $(20, 17, 14, 11, 8)$ based on start, stop, and step arguments. Normally, we use `for i in range(20, 5, -3)`, where i iterates through the sequence $(20, 17, 14, 11, 8)$ and the loop will be executed 5 times. In this program however, `(20, 5, -3)` is a tuple. i directly iterates through the tuple $(20, 5, -3)$ with 3 elements. Therefore, the loop will be executed 3 times as there are 3 elements in the tuple.

5	B	The core of the program is the function `g(n)`, which updates `n` based on its value modulo 5. Let's trace the change in `n` and the value of `n % 5` for the first few iterations:
		Initial: `n = 2025`, `n % 5 = 0`
		`i = 0`: `n = 2032`, `n % 5 = 2`
		`i = 1`: `n = 2029`, `n % 5 = 4`
		`i = 2`: `n = 2041`, `n % 5 = 1`
		`i = 3`: `n = 2037`, `n % 5 = 2`
		After the first iteration, the value of `n % 5` follows a repeating cycle of three states: $\mathbf{2} \to \mathbf{4} \to \mathbf{1}$. The total change to `n` over one full cycle is $(-3) + 12 + (-4) = 5$.
		The final value is the result of the first step, plus the change from 674 full cycles, plus the change from the final 2 steps, which is $2032 + (674 \times 5) + (-3) + 12 = 5411$.

Section B

Q	Answer and Explanation
A	`-4`
	`range(start, stop, step)` generates a sequence of integers starting from `start` up to but **excluding** `stop`, incremented by `step`. Here, `range(4, 0, -1)` means start at 4, decrease by 1 each step, and stop before reaching 0, generating the sequence $(4, 3, 2, 1)$.

Q	Answer and Explanation
B	`82`
	`for i in range(1,10)` will let iterate `i` through the sequence (1, 2, . . . , 9). The function `abs()` returns the absolute value of the specified number. So `abs(arr[i] - arr[i - 1])` returns the absolute difference between `arr[i]` and `arr[i-1]`. `tmp` saves the maximum value among all values of `x`, which are {`79,6,80,48,34,6,60,57,82`}. Hence, the maximum value 82 will be displayed.
C	`27`
	`append(x)` will add an element `x` to the end of the list. `pop()` will remove the element at the end of the list and return that element. In function `b(s)`, the last two elements of `s` are removed and saved as `x` (the last element) and `y` (the second to last element) respectively. After that, `x` is added to the end of `s` and the function returns the value of `y`. Actually, `b(s)` removes the second to last element and returns its value. The value of `s` and `total` change as following.

Instruction	`s`	`total`
`s.append(2)`	`[2]`	`0`
`s.append(14)`	`[2,14]`	`0`
`s.append(9)`	`[2,14,9]`	`0`
`s.append(11)`	`[2,14,9,11]`	`0`
`s.append(16)`	`[2,14,9,11,16]`	`0`
`total += b(s)`	`[2,14,9,16]`	`11`
`total += b(s)`	`[2,14,16]`	`20`
`s.pop()`	`[2,14]`	`20`
`s.append(7)`	`[2,14,7]`	`20`
`len(s) = 3 >= 2`		
`total += s.pop()`	`[2,14]`	`27`
`b(s)`	`[14]`	`27`
`len(3) = 1 < 2`		

Q	Answer and Explanation
D	`a[i]!=a[i-1]`
	The final merged list can be constructed in the following way: starting from the 1st element in the original list and iterate rightwards, the next element keeps getting merged to the previously iterated element until they are not equal, then we put latest one into the final list and move on, increasing the length of the final merged list by one. From this, we have `find_merged_length(a)` equal to 1 (the leftmost element) plus number of adjacent pairs with unequal elements in the original list.
E	`s="H"+s[1:]`
	Python strings cannot be changed — they are immutable. Therefore, assigning to an indexed position in the string (e.g. `s[0]='H'`) will result in an error. Instead, we need to create a new string. This can be done by slicing off the first character from the original string (`s[1:]`) and concatenate it to the back of `"H"`.
F	`0<=ni<n and 0<=nj<m`
	`(ni, nj)` iterates through all adjacent cell indices, so we have to check that `(ni, nj)` is within the range of the list of size $n \times m$. Note that this approach counts all adjacent pairs twice, so the answer is divided by 2.

Section C

Q		Answer and Explanation
G		`a[0]==a[-1] // a[0]==a[n-1]`
		By sorting the list, the minimum and maximum values are moved to the first and last positions. A check between `a[0]` and `a[-1]` is therefore sufficient, as if the smallest and largest elements are equal, all elements must be equal.
H		`a[0]`
		To verify all elements are the same, the loop must compare every element `x` against a single, constant reference. The first element, `a[0]`, is the only logical choice to serve as this required reference for the boolean check, as it's guaranteed to exist and represents the value that all other elements must match.
I	I1	`x`
	I2	`1`
		The function's strategy is to track the count `c` of a continuous sequence of identical values `v`. It will only return true if a single sequence spans the entire list. The else block executes whenever an element `x` breaks the current sequence. To handle this, the function must begin tracking a new sequence. It does this by setting `v` to the new value `x` and resetting the sequence count `c` to 1.
J		`a[x]+a[y]<=w`
		The total weight of box `x` and box `y` is `a[x]+a[y]`. To put both boxes onto the boat, we would need the total weight of box `x` and box `y` not exceeding `w`.
K	K1	`l<=r`
	K2	`l+=1` (can be interchanged with K3)
	K3	`r-=1` (can be interchanged with K2)
	K4	`r-=1`
		The algorithm can be described as follows: If we can put both box `l` and box `r` together on a ship, then we put them on the same ship. Otherwise, box `r` cannot pair up with any other box, and we should put it alone on a ship. Repeat the process until all boxes are handled. It might not be apparent why this produces the minimum number of round trips. We can think recursively: Let `min_rounds(l, r)` be the minimum number of round trips needed for boxes `l` to `r` inclusive. • When `a[l] + a[r] <= w`, we want `min_rounds(l, r) = min_rounds(l + 1, r - 1) + 1`. Suppose otherwise that `min_rounds(l, r) >= min_rounds(l + 1, r - 1) + 2`, then boxes `(l, x)` and `(r, y)` are paired up for some `l < x, y < r`. That means `a[l] + a[x] <= w` and `a[r] + a[y] <= w`. However, `a[r] + a[y] <= w` also implies `a[l] + a[r] <= w` and `a[x] + a[y] <= w`, i.e., we can pair up `(l, r)` and `(x, y)` instead. This implies `min_rounds(l + 1, r - 1) >= min_rounds(l, r) - 1`, and hence `min_rounds(l, r) = min_rounds(l + 1, r - 1) + 1`. • Otherwise, when `a[l] + a[r] > w`, clearly we cannot pair box `r` with any other box, and so `min_rounds(l, r) = min_rounds(l, r - 1) + 1`.

Paper 2 (C++)

Section A

Q	A	Explanation
1	D	`x += 1`, `x ^= 2`, and `x %= 3` are arithmetic operations that may affect the value of variable `x`. `x != 4` is a comparison operation meaning "`x` not equal to 4", and will never affect the value of `x`.
2	B	Overflow error occurs when the result of an arithmetic operation is too small or too large to be represented by the underlying data type. As 2147483648 exceeds the range of a 32-bit integer (from −2147483648 to 2147483647), overflow error occurs.
3	A	`x` doubles in each iteration in the loop, calculating 2 to the power of `i`. `x` is added to `s` when `i` is a multiple of 5, so `s` is the sum of 2 to the power of 0, 5, 10, and 15.
4	D	The sequences of x and y during the loop are $$\begin{array}{c\|cccccccc} i & 0 & 1 & 2 & 3 & 4 & 5 & 6 & \dots \\ \hline x & 0 & 3 & 2 & 0 & 3 & 2 & 0 & \dots \\ y & 1 & 4 & 6 & 5 & 4 & 6 & 5 & \dots \end{array}$$ They form cycles of length 3, so the contribution to `ans` repeats in a cycle $(-1, -4, -5)$ too. Since there are $\lfloor \frac{2024}{3} \rfloor = 674$ full cycles followed by 2 iterations, the final value of `ans` $= 674 \times (-10) + (-1) + (-4) = -6745$.
5	D	The prefix increments are first executed before the variables are used as parameters for the `f`() function. By calling `f`$(21, 31)$, since both variables `x` and `y` are passed by reference, the actual values of `x` and `y` become $21 + 2 = 23$ and $31 + 3 = 34$ respectively.

Section B

Q	Answer and Explanation
L	`16`
	Note that `j < i - j` is equivalent to `j + j < i` here. So for odd `i`, `ans` is increased `(i + 1) / 2` times; and for even `i`, `ans` is increased `i / 2` times.
M	`23`
	The program counts the number of integers within 1 to 77 that are multiples of 7 or contains the digit 7. There are 11 multiples of 7 and 15 integers that contain the digit 7. As 3 numbers (7, 70, 77) are both a multiple of 7 and contain the digit 7, we should subtract 3 from the count for a total of 23.
N	`1213121`
	The stack performs a depth-first process. For each positive `x`, it pushes `x - 1`, `-x`, `x - 1`; negatives are printed as `-x` (i.e., print the positive value). This yields the pattern for `x = 3`: `1213121`.
O	`break // i=6` Or any other reasonable answers
	Replacing the blank with `break` ensures the loop stops immediately when the first matching lesson is found, so `ans` contains the earliest common lesson.
P	`a[i]>x`
	As `a` is sorted in ascending order, for any indices `i, j`, `i < j` must hold if `a[i] < a[j]`. Hence we only need to find the element with the smallest index that is larger than `x`. This can be done by iterating `i` from `n - 1` to `0`, and updating `k` whenever `a[i] > x`.
Q	`a[n-i]>=i`
	Since `a` is sorted in ascending order, `a[n-i] >= i` if and only if the last `i` elements in `a` are `>= i`. Therefore, H-index can be computed by finding the largest `i` satisfying `a[n-i] >= i`.

Section C

Q	Answer and Explanation	
R	R1	`x==-1||a[i]>a[x] // a[i]>=a[max(0,x)]`
	R2	`x=i`
	We iterate through the array and check if the current index `i` has a value larger than the index `x` with the maximum element so far. If so, we update `x` to be `i`. However, as `x` is initially `-1`, we cannot perform a direct comparison for the first iteration, as it would access an element that is out-of-bound, causing a runtime error. The first answer in `R1` utilizes short-circuit evaluation to prevent this, while the second answer sets the compared target to `a[0]` when `x == -1`, which will only occur when `i == 0` and hence not affecting the answer.	
S	S1	`i>=k // i<n-k`
	S2	`a[FindMax(a)]`
	S3	`FindMax(a)`
	The `k`-th smallest element in `a` is the `(n-k+1)`-th largest element in `a`. We can repeatedly find the largest element in `a`, remove it (by setting it to 0) for `(n-k)` times. The next `FindMax` call will return the index of the `(n-k+1)`-th largest element in the original array.	
T	T1	`len-1`
	T2	`i-len+1`
	This program implements the Sliding Window algorithm to iterate through all continuous windows of `len` data points. To achieve this, `sum` maintains the running sum of the current window. Once `i` reaches `len-1`, `sum` contains the sum of the first `len` data points, so it is compared with `max_sum` to update the maximum sum found if necessary. Then, the data point at index `i-len+1` is subtracted from `sum` to shift the sliding window for the next iteration.	
U	`(double) // 1.0*`	
	Since both the returned value of `MaxSum` and `len` are integers, the program would perform an integer division by default. To return the floating point result of the division, we can add `(double)` to explicitly cast the returned value of `MaxSum`, or `1.0*` to convert it to a double value. The program will perform a floating point division for both methods.	

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 8.5 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	1
1	2
1.5	3
2	3
2.5	3.5
3	4
3.5	4
4	4.5
4.5	5
5	6
5.5	6.5
6	6.5
6.5	7
7	7
7.5	7.5
8	7.5
8.5	8
9	8
9.5	8.5
10	8.5

Raw Marks	Equated Marks
10.5	9
11	9.5
11.5	10
12	10.5
12.5	11
13	11.5
13.5	12.5
14	13
14.5	13
15	13.5
15.5	14
16	14.5
16.5	15
17	15.5
17.5	16
18	16.5
18.5	17
19	17.5
19.5	18
20	20