Hong Kong Olympiad in Informatics 2022/23
Heat Event (Junior Group)
Official Solution

Statistics (N = 350)
Full mark = 45. Maximum = 42. Median = 14. Advance to Final = 19 marks or above.

Section A

Q	A	Explanation
1	F	C++ supports Implicit Conversions. For example, if an `int` variable compares against a `float` variable, the `int` variable will be implicitly converted to a `float` first, then they are compared as `float` variables.
2	F	-12 is a counterexample.
3	T	A property of bitwise XOR is that the effect of xor-ing a number is canceled by performing the same action again: x `XOR` $z = y$ x `XOR` z `XOR` $x = y$ `XOR` x $z = y$ `XOR` x Hence, there always exist one (and only one) integer z that satisfies the requirement.
4	T	By birthday paradox formula $P(N) = \left(\frac{1}{365}\right)^N \times \frac{365!}{(365-N)!}$, the probability when `N = 50` is `97%`, which is much higher than `50%`.
5	T	For all odd integers $N > 1$, the statement is true because $N^2 + \left(\frac{N^2-1}{2}\right)^2 = \left(\frac{N^2+1}{2}\right)^2$. By doubling the side lengths of these triangles, we get that the statement is true for all even $N > 4$. Lastly, the statement is clearly true when $N = 4$.
6	D	In `5/4`, both numbers are in `int` type, so it performs integer division and the result is also in `int` type, therefore the first number outputted is 1. In `5.0/4`, `5.0` is in `double` type, so it performs float division and the result is in `double`, therefore the second number is 1.25. By the same reasoning, the third number is 1.25
7	B	`c = c + 1` is executed 5 times. `i` is incremented to 6 at the end of the loop, breaking the loop condition and hence ending the loop.
8	A	As the initial and final values are the same, there would be exactly 2 '+1' and 2 '-1' operations, resulting $\frac{4\times3}{2} = 6$ ways. However, notice that '-1' '-1' '+1' '+1' would lead to out of the bound, so there are $6 - 1 = 5$ valid ways.
9	C	When `y` is 5, `x == y` returns `true`, meaning that the ternary operator will return the first expression: `x + y`, which evaluates to 10. When `y` is 7, `x == y` returns `false`, meaning that the ternary operator will return the second expression: `x - y`, which evaluates to -2.
10	D	In C++, one cannot add or multiply an `std::string` and an integer. The only arithmetic operator defined for `std::string` is `operator+` which requires both operands to be of the type `std::string`, and there does not exist an implicit conversion from an integer to a `std::string` (one needs to call `std::to_string`). Similarly, there does not exist an implicit conversion from an `std::string` to an integer (one needs to invoke `std::stoi`) so the normal operators `operator+` and `operator*` of integers are not candidates. Thus, both programs do not compile.

<table>
<tr><th>Q</th><th>A</th><th>Explanation</th></tr>
<tr><td>11</td><td>C</td><td>Let N be the number of players in each game. The probability of being an imposter in each round is $2/N$. Adding the constraint that Alice is expected to be the imposter in at least one out of seven rounds, we have $2/N \geq 1/7$, which gives $N \leq 14$
Notice that the question asked for the "maximum number of friends to invite" instead of the "maximum number of players in the game", we need to deduct Alice from the maximum number of players.
Hence, the answer is $14 - 1 = 13$.</td></tr>
<tr><td>12</td><td>B</td><td>First, notice that the super-swap only acts on elements of the same index parity. It can be viewed as swapping to sort the odd elements and even elements independently.
i. Sort the odd and even elements and put them back in order, the array is $[2, 1, 4, 3, 6, 5, 8, 7]$, which is still not sorted. You may also observe that there is no way to swap element 1 to the front.
ii. The array can be sorted. The number of swaps required to sort a descending array to ascending is $n(n-1)/2$. So the answer is $4(4-1)/2 + 5(5-1)/2 = 16$.</td></tr>
<tr><td>13</td><td>B</td><td>The `break` statement only breaks the inner for-loop when $i = j$, the value of `cnt` can be calculated by $0 + 1 + 2 + ... + 9 = 45$.</td></tr>
<tr><td>14</td><td>C</td><td>i. Correct. You can compare the given string and the current `mid` element lexicographically, to determine whether the string is located in the left half or right half of the array.
ii. Correct. As long as you can determine whether the search target exists in the left half or right half of the array by comparing it to the `mid` element, you can binary search. Having multiple elements with the same value would not affect this.</td></tr>
<tr><td>15</td><td>B</td><td>This problem defines the rules of a context-free grammar. All well-formed parentheses matching could be generated by these actions.
i. `S → SS → (S)(S) → ()((S)) → ()(())`
ii. It cannot be generated using the given actions.
iii. `S → (S) → (SS) → ((S)(S)) → (()())`</td></tr>
<tr><td>16</td><td>A</td><td>We can first consider the first few elements:
<table>
<tr><td>i</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>$\cdots$</td></tr>
<tr><td>residue of `a[i]` when divided by 4</td><td>3</td><td>0</td><td>3</td><td>3</td><td>2</td><td>1</td><td>3</td><td>0</td><td>$\cdots$</td></tr>
</table>
We can observe that the pattern repeats, cycling through these 6 residues: 3, 0, 3, 3, 2, 1. This pattern repeats exactly 5 times in the first 30 elements. However, `a[1]` is not checked as `i` starts from 2, so the answer should be $5 - 1 = 4$.

Note that in C++, taking modulo of negative numbers gives a non-positive value (e.g. `-1 % 4` gives -1 instead of 3). However, this does not concern us in this question as we only need `a[i] % 4 == 0`, which holds for both positive and negative multiples of 4.</td></tr>
<tr><td>17</td><td>B</td><td>i. Correct. David may get rank 1 if he won both his games and Charlie vs Edward is a draw.
ii. Correct. David may get rank 1 if he won both his games and Charlie won against Edward or Edward won against Charlie.
iii. Incorrect. It is impossible for David to get rank 3.</td></tr>
<tr><td>18</td><td>C</td><td>Both `x` and `y` are pushed with the same elements in the same order, but since `x` is a stack while `y` is a queue, the elements to be compared and popped are the last and the first respectively in the order of pushing.
The loop terminate when either one of `x` and `y` is empty, and the final state of `x` is `{3, 1}` in the order of pushing, while `y` is empty.</td></tr>
<tr><td>19</td><td>D</td><td>Since 2023 different valid inputs do not cover all (valid or invalid) inputs, we cannot determinedly prove any of the 3 statements to be true.</td></tr>
</table>

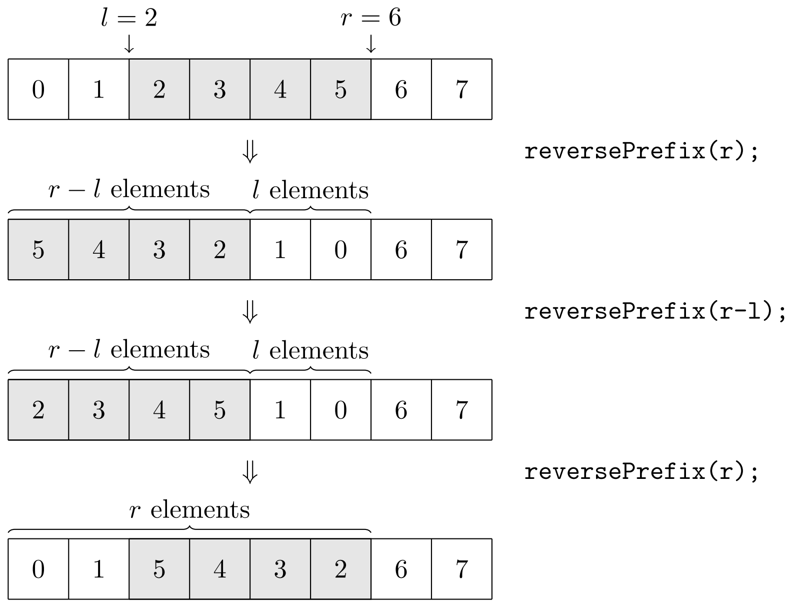
Q	A	Explanation
20	B	Since $0\|8 = 8, 1\|8 = 9, 2\|8 = 10, \ldots, 7\|8 = 15$, summing up $(0\|8, 1\|8, \ldots, 15\|8)$ is equal to summing up 8 to 15 twice, which is $(8+15) \times 8 = 184$.
21	D	Track the value of `x` in each loop: <table><tr><td>cnt</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td></tr><tr><td>x</td><td>28</td><td>24</td><td>20</td><td>23</td><td>22</td><td>18</td><td>14</td><td>10</td><td>13</td><td>12</td><td>8</td><td>4</td><td>0</td></tr></table>The return value of `f(28)` is therefore 12.
22	B	Consider a linear ordering of the circle and place Alice at the front, i.e. `AXXXXXXXX`. This covers all possible configurations as the circle is equivalent under rotation. Now we consider all possible positions of Bob, Bob cannot be on the left / right of Alice, i.e. `ABXXXXXX` and `AXXXXXXB`. This left us with 5 positions to place Bob at. Filling in the rest arbitrarily with other people, we have $5 \times (8-2)! = 3600$ distinct configurations
23	D	For a fixed `i`, there are $\lfloor \frac{n}{i} \rfloor$ different `j` such that the counter increases by 1. Hence, the code calculates the sum of $\lfloor \frac{n}{k} \rfloor$ for $1 \le k \le n$. Notice that for large k, the value of $\lfloor \frac{n}{k} \rfloor$ seldomly changes. Therefore, the answer is $50 + 25 + 16 + 12 + 10 + 8 + 7 + 6 + 5 \times 2 + 4 \times 2 + 3 \times 4 + 2 \times 9 + 1 \times 25 = 207$.
24	C	i. True. The statement "If Charlie plays football with friends or Charlie wins a chess game, Charlie feels happy" is equivalent to "If Charlie does not feel happy, Charlie does not play football with friends and Charlie does not win any chess game". Since there is no typhoon approaching Hong Kong, Charlie does not feel happy. Hence, he does not win any chess games. ii. True. Since there is no typhoon approaching Hong Kong, Charlie does not feel happy. Hence, he does not play football with friends. iii. False. If Charlie wins a chess game, he must feel happy. Therefore, if he does not feel happy, he must not win any chess games.
25	D	`v` is nice if there is between 4 and 8 (inclusive) numbers in `a` being larger than or equal to `v`. The numbers in `a` sorted in descending order are {`234, 123, 77, 56, 19, 4, -2, -15, -80, -147`}. Therefore, the range for which v is nice is $56 \ge$ `v` > -80.

Section B

Q	Answer and Explanation
A1	`text[i] == '/' //` `text[i] == 47`
	Locate the slash (`/`) in `text`. Alternatively, note that the ASCII code of slash is `47`, so it is possible to compare each `char` in `text` and `47` with implicit conversion.
A2 A3	`i-1, i+2 //` `i+2, i-1`
	`3` and `4` in `2023/24` are located in 1 index behind and 2 indexes in front of the slash (`/`) respectively. `3` and `4`'s ASCII codes decremented by 1 are the ASCII codes of `2` and `3` respectively.

B	`k/2, k-1-i //`
	`(k+1)/2, k-1-i //`
	`k-1-i, k-1-i //`
	`k-i, k-1-i //`
	`k-1, --k //`
	`k, --k`

	For the first half of the array, we will need to swap each element with the corresponding element in the second half of the array. Note that the array is 0-based.

Q	Answer and Explanation
C	`r, r-l, r`
	The reverse subarray function works as follows: 
D	`~x+1 // ~0*x`
	Integers are saved as 2's complement. A way to negate an integer is, to flip every bit of it, then add 1 to it. In `C++`, bitwise negation can be done with the operator `~`. Therefore the answer is `~x+1`. Alternatively, notice that the bitwise negation of 0 is -1, so you multiply it with x to negate x.
E	7
	The code calculates the number of triples `(i, j, k)` with `i < j` such that the `k`-th character of `i`-th string equals to the `k`-th character of `j`-th string. We may calculate it directly by enumerating over all pairs of strings.

F	`ab ab ab ab ac ac ac ac //`
	 `ab ab ab ab ab a axy axy //`
	 `abx abx abx aby aby ab a //`
	 `abc abc abc abc abc ab //`
	 `abcd abcd abcd abcd abcd`

	Notice that if there are `n` strings such that their `i`-th characters are the same, its contribution to the counter is `n * (n - 1) / 2`. We may use this observation to reduce the length of the strings.
G1	0
	Alice loses no matter what her first move is. One may enumerate all valid sequences of operations to verify this.
G2	3
	After Alice paints cell 3 black, Bob can paint either cell 1 or cell 5 black. In response, Alice can paint cell 5 or cell 1 black respectively, winning the game.
H1	5
	Alice may paint the cell at the middle black. Then, whatever Bob responses, Alice may colour the cell at the mirror position black, and Alice would win.
H2	7
	The argument in H1 can also be applied here.

Q	Answer and Explanation
I	`x%4!=2&&y%4!=2 // (x%4-2)*(y%4-2)`
	If the cell at (i, j) is a `'.'`, $i \not\equiv 2 \pmod 4$ and $j \not\equiv 2 \pmod 4$.

i\j	0	1	2	3	4	5	6	7	8
0	.	.	#	.	.	.	#	.	.
1	.	.	#	.	.	.	#	.	.
2	#	#	#	#	#	#	#	#	#
3	.	.	#	.	.	.	#	.	.
4	.	.	#	.	.	.	#	.	.
5	.	.	#	.	.	.	#	.	.
6	#	#	#	#	#	#	#	#	#
7	.	.	#	.	.	.	#	.	.
8	.	.	#	.	.	.	#	.	.

Q	Answer and Explanation
J	`x%8&&y%8 // x%8*(y%8) // x&7&&y&7`
	If the cell at (i, j) is a `'.'`, $i \not\equiv 0 \pmod 8$ and $j \not\equiv 0 \pmod 8$.

i\j	0	1	2	3	4	5	6	7	8
0	#	#	#	#	#	#	#	#	#
1	#	.	.	.	.	.	.	.	#
2	#	.	.	.	.	.	.	.	#
3	#	.	.	.	.	.	.	.	#
4	#	.	.	.	.	.	.	.	#
5	#	.	.	.	.	.	.	.	#
6	#	.	.	.	.	.	.	.	#
7	#	.	.	.	.	.	.	.	#
8	#	#	#	#	#	#	#	#	#

Q	Answer and Explanation
K	`(x==4)==(y==4) // (x==4)^(y!=4) // !(x==4^y==4)`
	If the cell at (i, j) is a `'.'`, either both $i \neq 4$ and $j \neq 4$, or both $i = 4$ and $j = 4$.

i\j	0	1	2	3	4	5	6	7	8
0	.	.	.	.	#	.	.	.	.
1	.	.	.	.	#	.	.	.	.
2	.	.	.	.	#	.	.	.	.
3	.	.	.	.	#	.	.	.	.
4	#	#	#	#	.	#	#	#	#
5	.	.	.	.	#	.	.	.	.
6	.	.	.	.	#	.	.	.	.
7	.	.	.	.	#	.	.	.	.
8	.	.	.	.	#	.	.	.	.

Q	Answer and Explanation

L	`(a-b+3)%3==1 //`
	`a-b%3==1 //`
	`(a+2)%3==b%3 //`
	`a-1==b%3 //`
	`(a-b+2)%3==0`

	Note that 1 beats 3; 2 beats 1; 3 beats 2. Arranging the numbers in a circle, a number beats the number two positions to its right. A natural way to model this cyclic relationship is modulo arithmetic; A beats B if and only if $A + 2 \equiv B \pmod 3$. Rearranging the expression gives the answer `(a-b+3)%3==1`. For someone from a more mathematically background, it may be unintuitive why the official solution involves a (+3), the necessity of the addition stems from subtleties of the modulo operator in C++; Such reader is encouraged to run the code without the addition to find out why.