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

Statistics (N = 415)
Full mark = 45. Maximum = 45. Median = 16.5. Advance to Final = 25 marks or above.

Paper 1 (Compulsory Part)

Section A

Q	A	Explanation
1	D	i. False. An `else` block is optional. ii. False. If the condition of the `while` loop is initially false, the loop body will not be executed at all. iii. False. If the condition of the `for` loop is initially false, the loop body will not be executed at all.
2	C	• Team A: Can win the tournament because they have a 50% chance to beat B, and if they advance, they have a 100% chance to beat C. • Team B: Can win the tournament because they have a 50% chance to beat A, and subsequently a 50% chance to beat C in the final. • Team C: Can win the tournament because they are guaranteed to beat D, and if B beats A, C has a 50% chance to defeat B in the final. • Team D: Is eliminated immediately because Team C has a 100% chance of defeating them in the first round.
3	A	Consider Bob's status: 1. If Bob is a student: Bob (a student) knows Charlie (a non-student). 2. If Bob is not a student: Alice (a student) knows Bob (a non-student). Hence, i. True. In both cases, a student knows a non-student. ii. False. In case 2, there is only one student, so there does not exist two students knowing each other. iii. False. In case 1, there is only one non-student, so there does not exist two non-students knowing each other.
4	C	A. False. The queue contains six integers, since both instances of 3 are stored. B. False. Since 1 was pushed first, it will be popped first, not 5. C. True. Popping the elements one by one results in $1, 3, 5, 3, 2, 4$ in that order, so 2 is popped before 4. D. False. After 6 is pushed, it is placed at the back of the queue. The next element popped is still 1, not 6.
5	D	There are only four possible numbers, namely $1, 2, 3, 4$. The following arrangements show that each of them can appear on the last card: • If the first three cards are Cards 1, 2 and 3 respectively, showing $1, 2, 3$, then the last card is Card 4, which can show either 1 or 4. • If the first three cards are Cards 4, 1 and 3 respectively, showing $1, 2, 3$, then the last card is Card 2, which can show either 2 or 3. Hence, any of the four numbers can appear on the front side of the last card, so the answer is 4.

Q	A	Explanation
6	C	`(A AND B) OR (NOT A AND C) OR (A AND NOT B)` ≡ `(A AND B) OR (A AND NOT B) OR (NOT A AND C)` (by commutative law) ≡ `(A AND (B OR NOT B)) OR (NOT A AND C)` (by distributive law) ≡ `(A AND TRUE) OR (NOT A AND C)` (by inverse law) ≡ `A OR (NOT A AND C)` (by identity law) ≡ `(A OR NOT A) AND (A OR C)` (by distributive law) ≡ `TRUE AND (A OR C)` (by inverse law) ≡ `A OR C` (by identity law)
7	C	Regardless whether `D` is a stack or a queue, the first and second values popped in the `pop()` call are 5 and 7 respectively.
8	D	Note that if there are two consecutive elements a_i and a_{i+1} such that $a_i < a_{i+1}$, then regardless of the operations performed, a_i is always strictly smaller than a_{i+1}. Hence, $a_i \geq a_{i+1}$ for all $0 \leq i < n-1$ is a necessary condition to make the elements equal. But since $a_5 < a_6$ in both arrays, we see that neither of the arrays can be made into equal elements.
9	A	Statement (i) is true. If Alice went to the party, by the first fact Bob will have gone to the party, and by the second fact Charlie did not go to the party, contradicting the assumption. Statement (ii) is false. We can only deduce that Bob did not go to the party. It is possible that all three of Alice, Bob and Charlie did not go to the party.
10	A	i. True. Each splitting step increases the total number of piles by exactly 1. Starting with 1 pile and ending with 15 piles requires exactly 14 steps. ii. True. Consider the last step of the process. The process stops only when all piles are of size 1. This means the final operation must have produced two piles of size 1, which can only be achieved by splitting a pile of size 2. Therefore, a pile of size 2 must have existed immediately before the last step. iii. False. We can prove this is false by constructing a counterexample. Start with the pile of 15 coins and split 1 coin out each step until the initial pile has 4 coins left. Then, split that into 2 coins + 2 coins. In this construction, we have avoided the pile of size 3.

Section B

Q	Answer and Explanation
A	`24`
	We would like to find the smallest t that is a multiple of 6, 8 and 12 simultaneously. In other words, $t = \text{lcm}(6, 8, 12) = 24$.
B	`2`
	Initially, $x = 0$. Then, the values of x after the first few iterations are listed as follows: i: 0 1 2 3 4 5 6 7 ... x: 1 3 2 0 1 3 2 0 ... Note that the value of x follows a $1, 3, 2, 0, ...$ cycle. The 2025-th element in this cycle is 1. So, the output is $a[1]$, which is 2.
C	`1/8`
	For `f()`&26 to be 0, the second, fourth and fifth least significant bits of the `f()` must be 0 when written in binary. Hence, the given program will output 0 if and only if `f()` returns 0 (00000_2), 1 (00001_2), 4 (00100_2), 5 (00101_2).

Q	Answer and Explanation
D	`14`
	Notice that the xor-sum of any number and itself is 0, and the xor-sum of `a` is 10. Therefore, `y` = `y^10^10` = `4^10` = 14.
E	`21`
	Since there are too many ways for arranging 1 and +, it would be time-consuming if we brute-force all the possibilities of putting 1 and +. Instead, we transform the task into thinking whether we can attain a particular value. First, note that if we write the final value as $\overline{D_n D_{n-1} \cdots D_0}$, then $1 \le D_n \le D_{n-1} \le \cdots \le D_0$, since we can only add expressions of all 1 together. Meanwhile, the minimum length of expression to attain that value is $(D_n + D_{n-1} + \cdots + D_0) + (D_0 - 1)$ since we need to add $(D_0 - 1)$ + signs. The possible 21 final values are listed as follows, grouped by their unit digits: 1. $D_0 = 1$ (7): 1, 11, 111, 1111, 11111, 111111, 1111111 2. $D_0 = 2$ (9): 2, 12, 112, 1112, 11112, 22, 122, 1122, 222 3. $D_0 = 3$ (4): 3, 13, 113, 23 4. $D_0 = 4$ (1): 4
F	`16`
	Only intersections between opposite-colour segments contribute to the cost. Hence each black-white intersection is counted *exactly once* (when the first of the two segments is removed), so the total cost is independent of the removal order and equals the number of intersections between black and white segments. Let's number the balls from 1 to 8 in clockwise order, with the topmost ball being 1. • The white line $(1, 4)$ intersects the black lines $(2, 5), (2, 6), (2, 8), (3, 5), (3, 6), (3, 8)$. (6 intersections.) • The white line $(4, 7)$ intersects the black lines $(2, 5), (2, 6), (3, 5), (3, 6), (5, 8), (6, 8)$. (6 intersections.) • The white line $(1, 7)$ intersects the black lines $(2, 8), (3, 8), (5, 8), (6, 8)$. (4 intersections.) Therefore, the answer is 16.

Section C

Q	Answer and Explanation	
G	G1	`10`
	G2	`1`
	The function `f(x)` needs to generate the next **repunit** (a number consisting only of 1s). If $x = 1$, the next is 11. $(1 \times 10 + 1 = 11)$. If $x = 11$, the next is 111. $(11 \times 10 + 1 = 111)$. The pattern to add a digit '1' to the end of a number is to multiply the number by 10 and add 1.	
H	H1	`d`
	H2	`x+r` // `d*r+r`
	The variable `d` holds the single repeating digit of `x` (e.g., if $x = 222$, $d = 2$). The `if` branch splits the problem into two cases. When $d < 9$, the next repdigit has the same length as x, but the digit is incremented by 1. Otherwise, the next repdigit is one digit longer than x with all 1s. The second case is handled by invoking the function `f(r)`, hence r should be the underlying repunit of x. (e.g. if $x = 222$, $r = 111$). So, `r = x / d`. The first case is easy to handle at this point. The next repdigit is simply the current x plus the repunit (`x + r`).	
I	`a+b>c`	
	It is given that `a` $<$ `b` $<$ `c`. Instead of checking for all 3 inequalities, we only have to check for `a` + `b` $>$ `c`, as both `a` + `c` $>$ `b` and `b` + `c` $>$ `a` are guaranteed to be true by `b` $<$ `c` and `a` $<$ `c` respectively.	
J	J1	`k<n`
	J2	`k-i-1`
	Note that `j` $<$ `i` $<$ `k`, so checking for all valid triples would require `k` $<$ `n`. For each `j`, `k` is the smallest value such that a_i, a_j and a_k do not form a valid triple, so all elements with indices between `i` and `k` exclusive are possible longest sides, hence we increment the answer by `k-i-1`.	

Paper 2 (Python)

Section A

<table>
<tr><th>Q</th><th>A</th><th>Explanation</th></tr>
<tr><td>1</td><td>A</td><td>In Python, the <code>sep</code> parameter in function <code>print</code> determines the separator between items when printing multiple items. For example, <code>print(1, 2, 3, sep="-")</code> results in printing <code>1-2-3</code>. However, in this case, there is actually only one item in <code>print</code>, which is the list. Therefore, the separator is not used and the list is printed as normal.</td></tr>
<tr><td>2</td><td>D</td><td>In Python, the <code><<</code> operator represents a bitwise left shift. This operation shifts the binary representation of a number to the left by a specified number of positions. For example, in $3 << 1$, the binary representation of 3 is 11_2. This operation shifts 11_2 to the left by 1 bit. The result is 110_2, which is 6 in decimal. In this question, the binary representation of 2 is 10_2. $2 << 4$ shifts 10_2 to the left by 4 bits. The result is 100000_2, which is 32 in decimal.</td></tr>
<tr><td>3</td><td>A</td><td>
<table>
<tr><th><code>i</code></th><th>Executed line</th><th><code>a</code></th></tr>
<tr><td>0</td><td><code>a[0] += a[1]</code></td><td><code>[3, 2, 3, 4, 5]</code></td></tr>
<tr><td>1</td><td><code>a[1] += a[2]</code></td><td><code>[3, 5, 3, 4, 5]</code></td></tr>
<tr><td>2</td><td><code>a[2] += a[3]</code></td><td><code>[3, 5, 7, 4, 5]</code></td></tr>
<tr><td>3</td><td><code>a[3] += a[4]</code></td><td><code>[3, 5, 7, 9, 5]</code></td></tr>
<tr><td>4</td><td><code>a[4] += a[0]</code></td><td><code>[3, 5, 7, 9, 8]</code></td></tr>
<tr><td>5</td><td><code>a[0] += a[1]</code></td><td><code>[8, 5, 7, 9, 8]</code></td></tr>
<tr><td>6</td><td><code>a[1] += a[2]</code></td><td><code>[8, 12, 7, 9, 8]</code></td></tr>
<tr><td>7</td><td><code>a[2] += a[3]</code></td><td><code>[8, 12, 16, 9, 8]</code></td></tr>
<tr><td>8</td><td><code>a[3] += a[4]</code></td><td><code>[8, 12, 16, 17, 8]</code></td></tr>
<tr><td>9</td><td><code>a[4] += a[0]</code></td><td><code>[8, 12, 16, 17, 16]</code></td></tr>
</table>
</td></tr>
<tr><td>4</td><td>D</td><td>In Python, floor division rounds down towards negative infinity, and the modulo result always has the same sign as the divisor. So the answer is <code>-2 1</code>.</td></tr>
<tr><td>5</td><td>D</td><td>The program performs a bubble sort for the array <code>a</code> and counts the number of swaps. Each time two adjacent elements are swapped, the number of inversions of the array decreases by 1. There will be no inversions for the final sorted array. So the answer is the number of inversions of the initial array, which is <code>15</code>.
(An inversion is a pair of indices (i, j) such that $i < j$ and <code>a[i]</code> $>$ <code>a[j]</code>.)</td></tr>
</table>

Section B

<table>
<tr><th>Q</th><th>Answer and Explanation</th></tr>
<tr><td>A</td><td>27</td></tr>
<tr><td></td><td>The loop index <code>i</code> starts from 1 and increases by 2 every time, until it is greater than or equal to the length of <code>a</code>. So the answer is equal to the sum of all elements with odd indices in the array <code>a</code>, which is $6 + 4 + 8 + 9 = 27$.</td></tr>
<tr><td>B</td><td>78</td></tr>
<tr><td></td><td>For each index <code>i</code>, the nested loop iterates through all indices <code>j</code> that are less than or equal to <code>i</code>. If the difference between <code>i</code> and <code>j</code> is even, the code adds <code>a[j]</code> to <code>b[i]</code>. Otherwise, it subtracts <code>a[j]</code> from <code>b[i]</code>. For example, when <code>i</code> $= 2$ and <code>j</code> $= 1$, the difference <code>i</code> $-$ <code>j</code> $= 1$ is odd. Therefore, <code>a[1]</code> is subtracted from <code>b[2]</code>. After all, the array <code>b</code> becomes $[28, -10, 22, 15, 8, 31, -16]$. The final value of <code>s</code> is the sum of all elements in array <code>b</code> $= 28 - 10 + 22 + 15 + 8 + 31 - 16 = 78$. Alternatively, observe that each <code>a[i]</code> contributes once to <code>s</code> when <code>i</code> is even, and doesn't contribute otherwise.</td></tr>
</table>

Q	Answer and Explanation
C	`28`
	Observe that `& 7` is equivalent to `% 8` in this context, and `dx ^= 1` toggles `dx` (makes it 0 if it is 1 and vice versa). Hence we get `a` as follows: 1 1 1 1 1 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 1 0 0 0 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 And the answer is just the total number of 1s in `a`.
D	`is_cube(n-i*i)` // `is_cube(n-i**2)`
	Since `a` and `b` are positive integers, if $a^2 + b^3 = n$, then $a \geq 1$ and $a^2 < n$. The variable `i` in the code iterates over exactly such values of `a`. Each `i` contributes 1 to the answer (stored using the variable `s`) if and only if there exists some `b` such that $b^3 = n - i^2$, or equivalently, $n - i^2$ is a perfect cube.
E	`"101", 3` // `'101', 3`
	In Python, multiplying a string by an integer repeats it. So `a="101"` and `b=3` gives `"101"*3 = "101101101"`.
F	`(C|D)^B` // `B^(C|D)` // `(D|C)^B` // `B^(D|C)`
	From the hint, the set of months containing the letter `r` are $\{1, 2, 3, 4, 9, 10, 11, 12\}$. Decoding the month number into the 4-tuple (A, B, C, D), we observe that the given set corresponds to the numbers such that: • $B = 0$, and either C or D is 1 (the numbers $1, 2, 3, 9, 10, 11$), or • $B = 1$ and $C = D = 0$ (the numbers $4, 12$). This corresponds to the boolean expression as shown.

Section C

Q	Answer and Explanation
G	`a[i]-a[i-1]`
	The list is beautiful if and only if the difference between consecutive elements are 1. If any one of the differences is not 1, the list is not beautiful.
H	`a[0]+i`
	The list is beautiful if and only if the i^{th} element equals to the sum of the first element and i. If any one of them does not equal to $a[0] + i$, the list is not beautiful.
I	`a[n-1]-a[0]==n-1` // `a[-1]-a[0]==n-1` // `max(a)-a[0]==n-1`
	The list is beautiful if and only if it is in increasing order, and the difference between the last and first element is $n - 1$, because this implies that the difference between consecutive elements are 1. If the list is in increasing order, but $a[n-1] - a[0]$ does not equal to $n - 1$, the list is not beautiful.
J	`x[-4:]`
	The required output is "Yes" if and only if the two words share the same last 4 characters.
K	`sorted(x)` // `set(x)`
	The required output is "Yes" if and only if the two words share the same (multi)set of characters.

Paper 2 (C++)

Section A

<table>
<tr><th>Q</th><th>A</th><th>Explanation</th></tr>
<tr><td>1</td><td>B</td><td>The first loop updates the array by adding the value of the previous element to the current element (calculating prefix sums):
• $i = 1$: <code>a[1]</code> becomes $1 + 2 = 3$
• $i = 2$: <code>a[2]</code> becomes $3 + 3 = 6$
• $i = 3$: <code>a[3]</code> becomes $6 + 4 = 10$
• $i = 4$: <code>a[4]</code> becomes $10 + 5 = 15$
The array is transformed into $\{1, 3, 6, 10, 15\}$, which is then printed by the second loop.</td></tr>
<tr><td>2</td><td>C</td><td>Assume the initial values of <code>a</code> and <code>b</code> are a and b respectively. The values of the variables after each line in each program segment are shown below:
i.
<table><tr><th></th><th>a</th><th>b</th><th>c</th></tr><tr><td><code>int c = a;</code></td><td>a</td><td>b</td><td>a</td></tr><tr><td><code>b = c;</code></td><td>a</td><td>a</td><td>a</td></tr><tr><td><code>a = b;</code></td><td>a</td><td>a</td><td>a</td></tr></table>ii.
<table><tr><th></th><th>a</th><th>b</th></tr><tr><td><code>a ^= b;</code></td><td>$a \oplus b$</td><td>b</td></tr><tr><td><code>b ^= a;</code></td><td>$a \oplus b$</td><td>a</td></tr><tr><td><code>a ^= b;</code></td><td>b</td><td>a</td></tr></table>iii.
<table><tr><th></th><th>a</th><th>b</th></tr><tr><td><code>a = a + b;</code></td><td>$a + b$</td><td>b</td></tr><tr><td><code>b = a - b;</code></td><td>$a + b$</td><td>a</td></tr><tr><td><code>a = a - b;</code></td><td>b</td><td>a</td></tr></table>So, only ii and iii correctly exchange the values of <code>a</code> and <code>b</code>.</td></tr>
<tr><td>3</td><td>B</td><td>The <code>+</code> operator behaves differently depending on the data type. For string <code>s1</code>, the operator performs concatenation, joining the string <code>"1"</code> and the character <code>'1'</code> to create <code>"11"</code>. For <code>s2</code> (which has type <code>char</code>) and <code>s3</code> (which has type <code>int</code>), the operator performs numeric addition using the ASCII value of <code>'1'</code>, which is 49. Therefore, <code>s2</code> = <code>s3</code> $= 49 + 49 = 98$. Since <code>s2</code> is a character type, it stores the character <code>'b'</code> (the character associated with ASCII code 98), while <code>s3</code> is an integer type and stores the numeric value 98.</td></tr>
<tr><td>4</td><td>B</td><td>The cast applies only to <code>a</code>, so <code>(int) a + b + c</code> evaluates to $1 + 1.7 + 1.7 = 4.4$. Finally, assigning it to <code>int d</code> truncates again: $d = 4$.</td></tr>
<tr><td>5</td><td>C</td><td>In C++, <code>^</code> is the bitwise XOR operator and <code>y ^= x[i]</code> is equivalent to <code>y = y ^ x[i]</code>, <code><<</code> is the bitwise left shift operator and <code>(y & (1 << i)) != 0</code> is true if the <code>i+1</code>-th least significant bit of <code>y</code> is 1. The values of <code>y</code> are 5, 23, 8, 6, and 22, and the first, second, and fifth values have their corresponding bit being 1.</td></tr>
</table>

Section B

<table>
<tr><th>Q</th><th>Answer and Explanation</th></tr>
<tr><td>L</td><td><code>18</code></td></tr>
<tr><td></td><td>After each statement:
<table>
<tr><th>Statement</th><th>Result</th></tr>
<tr><td><code>b *= c</code></td><td>$b = 3 \times 5 = 15$</td></tr>
<tr><td><code>a += b</code></td><td>$a = 2 + 15 = 17$</td></tr>
<tr><td><code>b -= c</code></td><td>$b = 15 - 5 = 10$</td></tr>
<tr><td><code>a /= c</code></td><td>$a = \left\lfloor \frac{17}{5} \right\rfloor = 3$</td></tr>
</table>
Therefore, the output is $3 + 10 + 5 = 18$.</td></tr>
<tr><td>M</td><td><code>21</code></td></tr>
<tr><td></td><td>When <code>i</code> $= 2$, <code>j</code> $= 4, 6, 8, 10, 12, 14$, so <code>i</code> is added to <code>ans</code> 6 times. When <code>i</code> $= 3$, <code>j</code> $= 9, 12, 15$, so <code>i</code> is added to <code>ans</code> 3 times. When <code>i</code> ≥ 4, <code>i</code>$^2 \geq 16$, the inner loop never runs. The final value of <code>ans</code> is $2 \times 6 + 3 \times 3 = 21$.</td></tr>
<tr><td>N</td><td><code>7</code></td></tr>
<tr><td></td><td>For an input n, let m be the output. Consider the binary representation of both of the integers. Observe that the i-th bit of m is set if and only if the i-th, $(i+1)$-th, $(i+2)$-th and $(i+3)$-th bits are all set in n. Since n has at most 6 set bits, we conclude that the program can output all values from 0 to 7 (inclusive), with the exception of 5. Considering the positions of the two set bits in the binary representation of 5, we require the first 6 bits in the input to be set, but that means the 1st bit of the output is also set, so an output of 5 is impossible.</td></tr>
<tr><td>O</td><td><code>x-y</code> // <code>x^y</code> // <code>y-x</code> // <code>y^x</code></td></tr>
<tr><td></td><td>When type casting from an <code>int</code> to a <code>bool</code>, 0 will be casted to <code>false</code> and all non-zero values will be cast to <code>true</code>. The expressions above all result in 0 if and only if x and y are equal, thus the function returns <code>true</code> when x and y are unequal.</td></tr>
<tr><td>P</td><td><code>ok*a[i]</code> // <code>ok&a[i]</code> // <code>ok&&a[i]</code></td></tr>
<tr><td></td><td>We leverage the fact that all elements in the array are either 0 or 1. If there is at least one 0 in the array, the logical/bitwise AND of the array becomes 0. Alternatively, the product of the array also becomes 0.</td></tr>
<tr><td>Q</td><td><code>1995+e<<'/'<<e-4</code></td></tr>
<tr><td></td><td>The clear way to output the HKOI's name from the 19^{th} HKOI onward would be <code>1997 + e - 2 << '/' << (1997 + e - 1) - 2000</code>. To cut the answer length down, we can omit the spaces and simplify the arithmetic expressions in the output.</td></tr>
</table>

Section C

Q		Answer and Explanation
R		`n%2==0&&m%2==0` // `n%2+m%2==0` // `!(n%2||m%2)`
		Both `n` and `m` must be even, so check that each has a remainder 0 when divided by 2.
S		`m=n;m<=k;m++` // `m=1;m<=n;m++` (or any reasonable answers)
		Since the pairs are unordered, `(n, m)` and `(m, n)` are the same, the loop should only consider $\texttt{m} \geq \texttt{n}$ (or $\texttt{m} \leq \texttt{n}$) to avoid double counting.
T		`n/2`
		For each even `n`, consider the good pairs `(m, n)` where $\texttt{m} \leq \texttt{n}$ and `m` is even. Since the even values between 1 and `n` are 2, 4, ..., `n`, there are exactly `n`/2 such values of `m`.
U	U1	`sum%3==0`
	U2	`a++` // `++a` // `a+=1`
		It is well known that an integer is divisible by 3 if and only if the sum of its digits is divisible by 3. The variable `sum` stores the digit sum from the `i`-th to the `j`-th character, inclusive. Since the two nested loops iterate over each pair of $0 \leq i \leq j < n$ exactly once, we can simply increment the variable `a` (which stores the answer) by 1 whenever the current digit sum is divisible by 3.
V		`a+=p[sum%3]`
		At the `i`-th iteration of the loop, `p[x]` stores the number of indices $0 \leq \texttt{j} \leq \texttt{i}$ such that the digit sum from the 0-th digit (inclusive) to the `j`-th digit (exclusive) is `x` modulo 3. Note that `s[j..i]` is divisible by 3 if and only if `s[0..j-1]` and `s[0..i]` have the same digit sum modulo 3. For each `i`, note that `p[sum%3]` counts the number of such valid `j` and is what we want.

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 12.5 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.

Python Version	C++ Version
0	0
0.5	1
1	2
1.5	3
2	4
2.5	5
3	5.5
3.5	6.5
4	7
4.5	8
5	8.5
5.5	9
6	9
6.5	9.5
7	10
7.5	10.5
8	11
8.5	11.5
9	12
9.5	12.5
10	12.5

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