Hong Kong Olympiad in Informatics 2018/19
Heat Event (Junior Group)
Official Solution

Statistics (N=256)
Full mark = 45. Maximum = 40. Median = 14.5. Advance to Final = 17 marks of above.

Section A1

Q	A	Explanation
1	T	The ASCII code of 'A' is 65 and that of 'a' is 97.
2	T	The size of a character variable is 1 byte, which is equal to 8 bits.
3	F	The expression i & -i returns the value of the least significant bit in i in its binary representation which can only be a power of two.
4	F	The name of a variable must contain only alphabets, numbers or an underscore ('_') and must not start with numbers.
5	F	Java is a language designed by James Gosling. JavaScript is a language designed by Brendan Eich. These two languages are not related.

Section A2

6	B	(n - n / i * i) is equivalent to (n % i), as (n / i) rounds down.
7	D	The program finds the sum of all positive odd numbers, subtracted by other numbers. Hence the sum = (7 + 9 + 11) - [(-3) + 2 + 6 + (-5) + (-10)] = 37.
8	D	Assigning a string to an integer variable is a syntax error that will be detected at compile time.
9	D	The function assigns a to result and multiplies a to result e times. Therefore, it returns the value a^{e+1}. Hence the value of my_pow(2, 10) = $2^{10+1} = 2^{11}$ = 2048.
10	D	f(1) + f(2) + f(3) + … + f(29) + f(30) = [f(0) + f(1) + f(2) + … + f(26)] + [f(27) + f(28) + f(29) + f(30)] Note that 0 = 000_3, 26 = 222_3. Hence f(0) + f(1) + f(2) + … + f(26) = 27 * 3 * (0 + 2) / 2 = 81. By 27 = 1000_3, 28 = 1001_3, 29 = 1002_3, 30 = 1010_3, we have f(1) + f(2) + f(3) + … + f(29) + f(30) = 81 + 1 + 1 + 1 + 1 + 2 + 1 + 1 = 89.
11	C	For a sum of some numbers to be odd, the number of odd numbers among the numbers must be odd. Inserting a plus sign after an even number does not change the count of odd numbers, while inserting a plus sign after an odd number increases the count of odd numbers by one. Among the first 9 digits, there are 6 even numbers and 3 odd numbers. Hence, the number of different expressions = $2^6 * ({}^3C_1 + {}^3C_3)$ = 64 * 4 = 256.
12	A	For a product to be an odd number, all the numbers multiplied must be odd numbers. Among the first 9 digits, the 2nd, 4th, 7th digit are odd numbers. The multiple signs can only be inserted after the 2nd, 4th and 7th digit. Therefore, there are 2^3 = 8 possibilities.
13	D	The for-loop finds the largest pair (i, j) such that i < j and a[i] = a[j], which is a[5] and a[7].
14	B	For (i), $11_6 = 7_{10}$, which is a prime number. Hence (i) is false. For (ii), consider the digits of a n-digit number. Note that $8 \equiv -1 \pmod 9$ and $8^2 \equiv 1$ (mod

<table>
<tr><td></td><td></td><td>9). Let the 0^{th} digit be the unit digit. As the i^{th} digit must be same as the $(n - 1 - i)^{th}$ digit, and i and (n - 1 - i) have different parity, the remainder when divided by 9 must be 0. Hence, all even-length palindrome in base 8 must be a factor of 9.</td></tr>
<tr><td>15</td><td>B</td><td>A single cycle of bubble sort is executed on a, with the original result stored in b.
For (i), consider b = {1, 0, 2, 3, …}. After the loop, a = {0, 1, 2, …}, where a[1] > b[1]. Hence, (i) need not be true.
For (ii), consider b = {0, 1, 3, 4, 2, 5, 6, …}. After the loop, a = {0, 1, 3, 2, 4, 5, …} with a[2] > a[3]. Hence, (ii) need not be true.
For (iii), as the last iteration of the loop swaps a[8] and a[9] if a[8] > a[9], a[8] <= a[9] must be true. Hence, (iii) must be true.</td></tr>
<tr><td>16</td><td>C</td><td>The combination that results in the minimum amount is hiring Bob, Daisy and Emily.</td></tr>
<tr><td>17</td><td>C</td><td>A pair of boolean expression is logically equivalent if they have the same truth table.
The truth tables are as follows:
<table>
<tr><td colspan="3"><code>x || (!x && y)</code></td><td colspan="3"><code>(x && !y) || y</code></td></tr>
<tr><td></td><td>x = 0</td><td>x = 1</td><td></td><td>x = 0</td><td>x = 1</td></tr>
<tr><td>y = 0</td><td>0</td><td>1</td><td>y = 0</td><td>0</td><td>1</td></tr>
<tr><td>y = 1</td><td>1</td><td>1</td><td>y = 1</td><td>1</td><td>1</td></tr>
<tr><td colspan="3"><code>(x || !y) ^ (!x || y)</code></td><td colspan="3"><code>(x && !y) || (!x && y)</code></td></tr>
<tr><td></td><td>x = 0</td><td>x = 1</td><td></td><td>x = 0</td><td>x = 1</td></tr>
<tr><td>y = 0</td><td>0</td><td>1</td><td>y = 0</td><td>0</td><td>1</td></tr>
<tr><td>y = 1</td><td>1</td><td>0</td><td>y = 1</td><td>1</td><td>0</td></tr>
</table>
From the truth tables, both pairs are logically equivalent.</td></tr>
<tr><td>18</td><td>A</td><td>The program repeatedly applies a permutation to an array that is initialized with a[i] = i.
The permutation contains two cycle: one of them is (2 → 0 → 6 → 1 → 3) and the other is (7 → 4 → 5).
For a[2], as 2 is in the first cycle and 2018 ≡ 3 (mod 5), the result should be 1.
For a[7], as 7 is in the second cycle and 2018 ≡ 2 (mod 3), the result should be 5.</td></tr>
<tr><td>19</td><td>D</td><td>f(x) = x AND x = x. g(x) = x XOR x = 0.
For (i), f(g(x)) = f(0) = 0. g(f(x)) = g(x) = 0 = f(g(x)). Therefore, (i) is true.
For (ii), f(f(x)) = f(x) = x = f(x). Therefore, (ii) is true.
For (iii), g(g(x)) = g(0) = 0 = g(x). Therefore, (iii) is true.</td></tr>
<tr><td>20</td><td>D</td><td>As Alice should not sit next to Bob, and Charlie must not sit next to Dave, Alice and Bob must sit in either the 1st and the 3rd seat or the 2nd and the 4th seat. With either Alice sitting before Bob or not and Charlie sitting before Dave or not, the number of combinations = 2 * 2 * 2 = 8.</td></tr>
<tr><td>21</td><td>C</td><td>An element in a stack can only be popped out when all elements pushed after it is popped. Therefore, a stack is ‘first in, last out’.
An element in a queue can only be popped out when all elements pushed before it is popped. Therefore, a queue in ‘first in, first out’.</td></tr>
<tr><td>22</td><td>B</td><td>Let the top left cell of the board be (0, 0). The ideal arrangement is to put a king in each cell (i, j) where both i and j are even. Hence there are (98/2 - 0 + 1)^2 = 50^2 = 2500 kings at most.</td></tr>
<tr><td>23</td><td>C</td><td>Let R be the statement “it is raining” and W be the statement “the ground is wet”.
For (i), the second statement is the contrapositive of the first statement, i.e. (R → W) = ¬W → ¬R. Therefore, they are equivalent.
For (ii), the first statement is (R → W). The second statement is (W or ¬R). (R → W) is only false when R is true and W is false. (W or ¬R) is also only false when R is true and W is false. Therefore, they are equivalent.</td></tr>
</table>

<table>
<tr><td>24</td><td>A</td><td>The increment statement is not inside any of the for loops and if statements. Therefore, it is only executed once.</td></tr>
<tr><td>25</td><td>C</td><td>The first set of for-loop initializes the array as follows:
(the first index is the row and the second index is the column)
<table>
<tr><td>0</td><td>1</td><td>2</td><td>3</td></tr>
<tr><td>4</td><td>5</td><td>6</td><td>7</td></tr>
<tr><td>8</td><td>9</td><td>10</td><td>11</td></tr>
<tr><td>12</td><td>13</td><td>14</td><td>15</td></tr>
</table>
After that, the second set of for-loops rotates the array clockwise by 90 degrees and results in the array as follows.
<table>
<tr><td>12</td><td>8</td><td>4</td><td>0</td></tr>
<tr><td>13</td><td>9</td><td>5</td><td>1</td></tr>
<tr><td>14</td><td>10</td><td>6</td><td>2</td></tr>
<tr><td>15</td><td>11</td><td>7</td><td>3</td></tr>
</table>
</td></tr>
</table>

Section B

<table>
<tr><th colspan="4">Answer and Explanation</th></tr>
<tr><td></td><td>Pascal</td><td>C</td><td>C++</td></tr>
<tr><td rowspan="2">A</td><td colspan="3"><code>10</code></td></tr>
<tr><td colspan="3">The array <code>a[]</code> is initialized to all 0s. For any number <code>j</code>, when a factor of <code>j</code> is found, <code>a[j]</code> is flipped. As only square numbers have an odd number of factors, only <code>a[i]</code> for square numbers <code>i</code> is 1, while all other <code>a[i]</code> are 0. There are 10 square numbers less than or equal to 100, which is the answer.</td></tr>
<tr><td>B</td><td><code>i := 1 to n</code></td><td><code>i = 1; i <= n; i++</code></td><td><code>i = 1; i <= n; i++</code></td></tr>
<tr><td>C</td><td><code>sum + i</code></td><td><code>sum + i</code></td><td><code>sum + i</code></td></tr>
<tr><td rowspan="2">D1, D2</td><td colspan="3"><code>16, 25, 36, 49, 64, 81 (any two)</code></td></tr>
<tr><td colspan="3">When <code>n</code> is a square number, √<code>n</code> is double counted.</td></tr>
<tr><td>E1</td><td><code>i*i=n</code>
<code>sqrt(n)=i</code>
<code>n/i=i</code></td><td><code>i*i==n</code>
<code>sqrt(n)==i</code></td><td><code>i*i==n</code>
<code>sqrt(n)==i</code></td></tr>
<tr><td rowspan="2">E2</td><td><code>sum-i</code></td><td><code>sum-i</code></td><td><code>sum-i</code></td></tr>
<tr><td colspan="3">To prevent double counting sqrt(n), subtract i from the sum.</td></tr>
<tr><td>F</td><td><code>sum_factors(n)</code></td><td><code>sum_factors(n)</code></td><td><code>sum_factors(n)</code></td></tr>
<tr><td>G1</td><td><code>sum>2*n</code></td><td><code>sum>2*n</code></td><td><code>sum>2*n</code></td></tr>
<tr><td>G2</td><td><code>sum=2*n</code></td><td><code>sum==2*n</code></td><td><code>sum==2*n</code></td></tr>
<tr><td>H</td><td colspan="3"><code>45</code></td></tr>
<tr><td>I</td><td colspan="3"><code>25</code></td></tr>
<tr><td rowspan="2">J</td><td colspan="3"><code>10 9 8 7 6 5 4 1 2 3</code></td></tr>
<tr><td colspan="3">This program finds the number of inversions in the array, the number of inversions is calculated by finding the number of pair(i, j) where i < j and a[i] > a[j].
To construct an array with 42 inversions, we start with a reversely sorted array, and we flip any 3 pairs of numbers which the pair is an inversion.</td></tr>
<tr><td>K</td><td colspan="3"><code>50</code></td></tr>
</table>

	Note that the value of the first two digits determines the last two digits in the password, there are at most 9 * 9 = 81 possible combinations. Let i and j be the first two digits. Both of the digits cannot be 0, so i * j must not be a factor of 10 and i * j must be larger than 10. By subtracting the number of pairs of (i, j) that contain 0, we can find the answer.		
L1	`a[m]>=target`	`a[m]>=target`	`a[m]>=target`
L2	`l:=m+1`	`l=m+1`	`l=m+1`
L3	`m-1`	`m-1`	`m-1`
	The program is a binary search. If the index is not found, m will be equal to 0 at the end of the while loop. If the index is found, m will be (the required index + 1). Therefore (m - 1) is the required index or -1 if it is not found.		
M	`2`		
	As the two if statements do not have brackets, the else statement follows the last if statement. Hence, any even number that is not a multiple of 3 will cause the program to output 'ODD'.		
N1	`17 // 18`	`46 // 47`	`76 // 77`
N2	`else // [space]else`	`else; // ;else`	`else; // ;else`
	It ends the inner if statement and adds an else statement to the outer if statement.		
O	`(E-S)div J+(E-S)mod J<=T`	`(E-S)%J+(E-S)/J<=T`	`(E-S)%J+(E-S)/J<=T`
	The program calculates the minimum required step to walk the distance (E-S), then checks whether it is less than or equal to T.		