Hong Kong Olympiad in Informatics 2020/21
Heat Event (Junior Group)
Official Solution

Statistics (N = 233)
Full mark = 45. Maximum = 40.5. Median = 18. Advance to Finals = 20.5 marks or above.

Section A1

<table>
<tr><th>Q</th><th>A</th><th>Explanation</th></tr>
<tr><td>1</td><td>F</td><td>A string is a sequence of characters. <code>HKOI21</code> and <code>HKOIXXI</code> contain 6 and 7 characters, so it requires $8 \cdot 6 = 48$ and $8 \cdot 7 = 56$ bits to store them respectively.</td></tr>
<tr><td>2</td><td>T</td><td>To determine whether a number is odd or even, we only need to consider the least significant bit, i.e. the right-most bit.
<table>
<tr><td><code>x</code></td><td><code>y</code></td><td><code>z</code></td><td><code>x & y</code></td><td><code>z | y</code></td><td><code>(x & y) ^ (z | y)</code></td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td><td>1</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>1</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>1</td><td>1</td><td>0</td></tr>
</table>
By considering all possible cases, we can see that the function returns an odd number if and only if <code>x</code> and <code>y</code> do not have the same parity.</td></tr>
<tr><td>3</td><td>T</td><td>For each digit in the base 8 representation of n, we may rewrite it as a 3 digit number in base 2. For example, 7 in base 8 is 111 in base 2. We can observe that for each base 8 digit (0 - 7), its digit sum in base 2 is always no larger than that in base 8. Therefore, we may conclude that the digit sum in base 2 is no larger than base 8.</td></tr>
<tr><td>4</td><td>F</td><td>By Pythagoras' Theorem, the side lengths of a right-angled triangle satisfy $a^2 + b^2 = c^2$.If $a, b, c$are positive odd integers, the left-hand side of the equation is an even number but the right-hand side is an odd number. Hence, there is a contradiction.</td></tr>
<tr><td>5</td><td>F</td><td>Colour the chessboard in black and white alternately as in a regular 8×8 chessboard. A domino on the board must cover one black and one white cell. As the two diagonally opposite corners are of the same color, the number of black and white cells is not equal after the removal. Therefore, we cannot cover the board with such dominoes.</td></tr>
</table>

Section A2

Q	A	Explanation
6	A	The binary representation of 8 is 1000, which has 1 ‘1-bit’.
7	D	Since -1 is negative and the number is represented in two’s complement, the sign bit (which represents -2^{31}) is set to 1. The remaining bits must sum to $2^{31} - 1$ so all 31 of them are set to 1. Thus, every bit of the signed 32-bit integer is 1.
8	C	By dry running the code, we find that `x` takes the values 16, 8, 4, 2, 1. Since `a` is initialized as `x`, the answer is $16 + 16 + 8 + 4 + 2 + 1 = 47$.
9	B	As there are only two iterations in the for loop, we may dry run the program in very little time.
10	C	C is the contrapositive of the given conditional statement, so it must be true. Note: In first order logic, “If A then B” is equivalent to “If NOT B then NOT A”. The statement “If NOT B then NOT A” is called the contrapositive of “If A then B”
11	D	Let n be the initial number of bottle caps she has. By trading a new bottle of juice, the value of n decreases by 1. Therefore, we may set up the equation $2n - 1 = 83$and solving the equation gives $n = 42$ as the solution.
12	A	`a[i]` stores the exponent of 2 in the prime factorization of i. As $2020 = 2^2 \times 5 \times 101$, the answer is 2.
13	C	Note that you cannot apply the square operation for more than 2 times. Consider the following 4 cases. Case 1: Apply +1 operation until it reaches 100. Case 2: Apply +1 operation until it reaches a number between 4 and 10. After squaring a number between 4 and 10, you may no longer perform another squaring operation. Therefore, the case contributes 7 ways to the answer. Case 3: Apply +1 until it reaches 2. After squaring, x becomes 4, you may either square it again, increment it until it reaches a number between 5 and 10, or increment it until 100. These 3 scenarios contribute 1, 6 and 1 ways respectively. Case 4: Apply +1 until it reaches 3. This case is similar to case 3, you may either square it again, increment it until 10, or increment it until 100. These 3 scenarios contribute 1, 1, and 1 way respectively. Therefore, the answer = 1 + 7 + (1 + 6 + 1) + (1 + 1 + 1) = 19.

14	D	For (i), we consider each bit separately: if `a` has a 1 bit, then `a or b` is 1, so `a` will not decrease.
		Similarly, for `(ii)`, if `a` has a 0 bit, then `a and b` is 0, so `a` will not increase.
		For (iii), `a xor b = a` if and only `b = 0`. As it is given that `b` is a positive integer, the statement must be true.

<table>
<tr><td>15</td><td>C</td><td>The inner loop replaces element <code>a[i]</code> with the maximum element in <code>a[i+1…n]</code>. In fact, this is an incomplete selection sort algorithm that orders the first 4 greatest elements of <code>a[1…9]</code> in descending order. Note that the number at index 0 is not considered. The final values of <code>a[1…4]</code> would be <code>{10, 9, 8, 6}</code> intuitively.</td></tr>
<tr><td>16</td><td>D</td><td>All of the given statements are true. In particular, a stack can be implemented by a singly linked list whose head is where the stack operations happen.</td></tr>
<tr><td>17</td><td>D</td><td>O represents an occupied seat, while X represents an unoccupied seat.
We may count all combinations by considering the following 3 cases.
Case 1: (the Os in between each X are the same)
XOOXOOXOO
OOXOOXOOX
Case 2: (some Xs are seating together)
OOXXOOXOO
OOXOOXXOO
Case 3: (neither Case 1 nor Case 2)
OXOXOOXOO
OXOOXOXOO
OXOOXOOXO
OOXOXOXOO
OOXOXOOXO
OOXOOXOXO
Answer: 10</td></tr>
<tr><td>18</td><td>B</td><td>For an even-length string, the answer is obviously $\frac{n}{2}$ which rules out option A and D. For an odd-length string, note that the middle character does not need to be compared with. Therefore, the answer is $\frac{n-1}{2}$, which is equivalent to option B.</td></tr>
<tr><td>19</td><td>C</td><td>Two boolean expressions are equivalent if they have the same truth table.

The truth tables are as follows:
<table>
<tr><td colspan="3"><code>(A OR B) AND (NOT A OR NOT B)</code></td><td colspan="3"><code>(A AND B) XOR (A OR B)</code></td></tr>
<tr><td></td><td>B = False</td><td>B = True</td><td></td><td>B = False</td><td>B = True</td></tr>
<tr><td>A = False</td><td>False</td><td>True</td><td>A = False</td><td>False</td><td>True</td></tr>
<tr><td>A = True</td><td>True</td><td>False</td><td>A = True</td><td>True</td><td>False</td></tr>
<tr><td colspan="3"><code>(A XOR B) OR (A AND B)</code></td><td colspan="3"><code>(A OR B) AND (A XOR B)</code></td></tr>
<tr><td></td><td>B=False</td><td>B=True</td><td></td><td>B = False</td><td>B = True</td></tr>
<tr><td>A = False</td><td>False</td><td>True</td><td>A = False</td><td>False</td><td>True</td></tr>
<tr><td>A = True</td><td>True</td><td>True</td><td>A = True</td><td>True</td><td>False</td></tr>
</table></td></tr>
<tr><td>20</td><td>B</td><td>By dry running the code, <code>res</code> will be 22 finally.</td></tr>
<tr><td>21</td><td>C</td><td>The function $f(x)$ counts the number of factors of x. As $2520 = 2^3 \cdot 3^2 \cdot 5 \cdot 7$, the answer would be $(3 + 1)(2 + 1)(1 + 1)(1 + 1) = 48$.</td></tr>
</table>

<table>
<tr><td>22</td><td>C</td><td>The code is an implementation of the Floyd cycle detection algorithm. After both while loops, the value of x would be equal to the starting point of the cycle, that is 6. The program outputs <code>a[6]</code> which points to 8.

You may also dry run the code to find the answer.</td></tr>
<tr><td>23</td><td>B</td><td>For (i), for k flips of a fair coin, there are a total of 2^k combinations, which is not a multiple of 3.
For (ii), we can split the outcome of the die into 3 sets: $\{1, 2\}$, $\{3, 4\}$, $\{5, 6\}$. By rolling the die twice, there are $3 \cdot 3 = 9$ distinct and equally likely combinations which can be used to generate an integer between 1 and 9.</td></tr>
<tr><td>24</td><td>B</td><td>We may exhaust Silloh's. Note that no boys can stand before Silloh, therefore the required probability = 1/7 × (1 + 3/6 + 3/6 × 2/5 + 3/6 × 2/5 × 1/4) = 1/4</td></tr>
<tr><td>25</td><td>D</td><td><code>a[i] * a[(i+2)%5]</code> is added to <code>c</code> <code>a[i]</code> times if and only if <code>a[i]</code> is positive.
<table>
<tr><td>Option</td><td>Input</td><td><code>c</code></td><td>Output</td></tr>
<tr><td>A</td><td>7 2 -3 5 6</td><td>$7 \cdot 7 \cdot (-3) + 2 \cdot 2 \cdot 5 + 5 \cdot 5 \cdot 7 + 6 \cdot 6 \cdot$</td><td>120</td></tr>
<tr><td>B</td><td>-1 6 0 3 2</td><td>$6 \cdot 6 \cdot 3 + 3 \cdot 3 \cdot (-1) + 2 \cdot 2 \cdot 6$</td><td>123</td></tr>
<tr><td>C</td><td>-1 3 7 8 2</td><td>$3 \cdot 3 \cdot 8 + 7 \cdot 7 \cdot 2 + 8 \cdot 8 \cdot (-1) + 2 \cdot 2 \cdot$</td><td>118</td></tr>
<tr><td>D</td><td>2 -1 5 -5 6</td><td>$2 \cdot 2 \cdot 5 + 5 \cdot 5 \cdot 6 + 6 \cdot 6 \cdot (-1)$</td><td>134</td></tr>
</table>
Hence, D gives the largest output.</td></tr>
</table>

Section B

<table>
<tr><th colspan="4">Answer and Explanation</th></tr>
<tr><th></th><th>Pascal</th><th>C</th><th>C++</th></tr>
<tr><td>A1</td><td colspan="3"><code>true</code></td></tr>
<tr><td>A2</td><td colspan="3"><code>true</code></td></tr>
<tr><td rowspan="2">A3</td><td colspan="3"><code>false</code></td></tr>
<tr><td colspan="3">Add every digit and check whether the sum is divisible by 3.</td></tr>
<tr><td rowspan="2">B</td><td colspan="3"><code>667</code></td></tr>
<tr><td colspan="3">$g(x)$ returns <code>true</code> when x is a multiple of 3. Hence, the answer $= 1000 - floor(1000/3) = 667$.</td></tr>
<tr><td>C</td><td colspan="3">
<table>
<tr><td></td><td><code>1</code></td><td><code>5</code></td><td><code>3</code></td></tr>
<tr><td><code>5</code></td><td><code>1</code></td><td><code>2</code></td><td><code>2</code></td></tr>
</table>
</td></tr>
</table>

3	0	2	1
1	0	1	0

Considering the first row, it is obvious that you must fill in 2 ‘2’s and 1 ‘1’. As the first column requires a sum of 1, a ‘2’ cannot be filled in. Filling in the remaining blanks is then straightforward.

D

	316	335	354
316	102	102	112
354	112	121	121
335	102	112	121

We can split this problem into three easy subproblems by considering each digit separately. The hundreds digit is trivial. The tens digits are a permutation of the numbers in the previous problem. The subproblem of ones digit can be solved by trial and error or by noting that 4, 5, 6 are equivalent to 1+3, 2+3, 3+3:

	3+3	2+3	1+3
3+3	1+1	1+1	1+1
1+3	1+1	0+1	0+1
2+3	1+1	1+1	0+1

We may obtain the final grid by concatenating the answers to the subproblems.
Actually, the problem is designed such that the hundreds and units digits will align considering only the tens digit.

E

`b+2 // a+5 // a*2`	`b+2 // a+5 // a*2`

Character `n` is at index 6, so the while loop should be executed exactly 6 times. An intuitive solution is to increment `b` by 2 every time.

F

6210001000	6210001000	6210001000

Here is one way of getting to the answer: we try putting 9 at the first digit (9000000000), then the last digit should become a 1 (9000000001). Now, a 1 should be put in the second digit (9100000001). Since there are 2 ones, we change the second digit to a 2 (9200000001) and subsequently the third digit to a 1 (9210000001). Note that there are six 0s so we update the first digit to a 6 (6210000001). Finally, observe that by changing the position of the last 1, we arrive at a valid number 6210001000.

G

125346	125346	125346

The given code is a faulty bubble sort algorithm. The easiest way to obtain the answer would be to dry run the code.

H

BABABA	BABABA	BABABA

<table>
<tr><td></td><td colspan="2">Observe that the faulty algorithm only works when there are no consecutive <code>A</code>s, therefore, the input <code>BABABA</code> would give the desired order after sorting.</td></tr>
<tr><td rowspan="2">I</td><td colspan="2"><code>4</code></td></tr>
<tr><td colspan="2">To determine whether $x$is prime, it is sufficient to check for divisors up to $\sqrt{x}$. However, the given function does not check $\sqrt{x}$. Hence, the function returns a wrong answer when $x$has exactly one divisor, $\sqrt{x}$, apart from 1 and itself, i.e. when $x = 2^2, 3^2, 5^2, 7^2$.</td></tr>
<tr><td>J1</td><td><code>20</code></td><td><code>48</code></td></tr>
<tr><td rowspan="2">J2</td><td><code>while(i*i<=x)do //
while(i<x)do //
while(i*i<x+1)do</code></td><td><code>while(i*i<=x) // while(i<x) //
while(i*i<x+1) //
while(i<=floor(sqrt(x)))</code></td></tr>
<tr><td colspan="2">By changing < into <=, the function now checks $\sqrt{x}$ too.</td></tr>
<tr><td rowspan="2">K</td><td><code>(j-i<=1)and(i-j<=1) //
abs(i-j)<2 //
(j-i)*(j-i)<=1 //
(j=i)or(j=i-1)or(j=i+1
)</code></td><td><code>j-i<2&&i-j<2 // abs(i-j)<2 //
(j-i)*(j-i)<=1 // j==i||j==i-1||j==i+1</code></td></tr>
<tr><td colspan="2">Note that the <code>#</code>s consist of 3 diagonal lines which can be described as <code>j-i = -1</code>, <code>j-i = 0</code> and <code>j-i = 1</code>.</td></tr>
<tr><td rowspan="2">L</td><td><code>(i+j)div 3=3</code></td><td><code>(i+j)/3==3 // (i+j)%12>8 //
(i+j+1)%10<3</code></td></tr>
<tr><td colspan="2">Similarly, note that the <code>#</code>s consist of 3 diagonal lines which can be described as <code>i+j = 9</code>, <code>i+j = 10</code> and <code>i+j = 11</code>.</td></tr>
<tr><td>M</td><td colspan="2">There are more than 1000+ solutions to this problem. One of the most intuitive solutions is <code>[[↘↗]4[↙↖]4]4</code>. The path of the bishop should resemble a zig-zag pattern. Note that a command is voided if it moves the bishop out of the chessboard.
Examples of other solutions: <code>[↗↘[↗]5[↙]6]8, [↗↘[↗]5[↙]6]7, [[↙↖]4[↘↗]4]4</code></td></tr>
</table>