Hong Kong Olympiad in Informatics 2019/20
Heat Event (Junior Group)
Official Solution

Statistics (N = 266)

Full mark = 42. Maximum = 41. Median = 11. Advance to Final = 15.5 marks or above.

Section A

Q	A	Explanation
1	F	Binary search can only be applied to a sorted array.
2	F	Although most computers have a single CPU, some motherboards had multiple CPU sockets to increase processing power when multi-core CPU was less popular.
3	F	The author dedicates his work to the public domain by waiving all his rights to the work under copyright law, so no attribution is needed.
4	T	2147483647_{10} = 01111111 11111111 11111111 11111111_2. After adding 10, it becomes 10000000 00000000 00000000 00001001_2 (-2147483639_{10}), an overflow error occurred. However, by subtracting 10 from it, the number would become 2147483647_{10} again, which is the correct result of the expression and can be stored in a 32-bit signed integer.
5	T	The sign of the result of a Mod operator is always the same as the sign of the left side operand. In fact, m mod n (C/C++: m % n) is equivalent to m - (m div n) * n (C/C++: m - (m / n) * n) in most programming languages (including Pascal and C/C++). Thus, the expression always evaluates to true.
6	B	`sum` stores the sum of all even numbers between 1 and 100. $2+4+6+...+100 = (2 + 100) * 50 / 2 = 2550$ `sum2` stores the sum of squares of all numbers (not even numbers) between 1 and 100. The sum of squares of first n natural numbers is n(n+1)(2n+1)/6 $1^2+2^2+3^2+...+100^2 = 100 * 101 * 201 / 6 = 338350$
7	D	The program calculates the frequency of digits 0-9, multiplied by the digit itself, appearing in array `a`, and stores them in `b[0]-b[9]` respectively. Then it outputs `b[3]` (2*3=6), `b[5]` (0*5=0) and `b[7]` (2*7=14).
8	D	The program finds the prime numbers between 2 and 100. `a[1]` is initialized with 1 and 97 is a prime number. So, both `a[1]` and `a[97]` are equal to 1.

9 C

Second number	#possible third number	#possible first number
2	**4** (3 4 5 6)	**1** (1)
3	**3** (4 5 6)	**2** (1 2)
4	**2** (5 6)	**3** (1 2 3)
5	**1** (6)	**4** (1 2 3 4)
6	**0**	**5** (1 2 3 4 5)

possible cases = 4 * 1 + 3 * 2 + 2 * 3 + 1 * 4 + 0 * 5 = 20

total cases = 6 * 6 * 6 = 216

Answer = 20 / 216 = **5 / 54**

10 C Consider the cases when Tom stands at the leftmost end of the line, as boys and girls must stand alternatively, the arrangement is as follows (T = Tom, G = Girl, B = Boy): TGBGBGBGBGBG, which is 5! * 6! = 86400.

When Tom stands at the rightmost end of the line, the arrangements are the reversed version of Tom standing at the leftmost end of the line. And thus, the answer is 86400*2 = 172800.

11 D Trace the program carefully:

i	q[0]	q[1]	q[2]	q[3]
push(1)	1	0	0	0
push(4)	1	4	0	0
push(3)	1	4	3	0
push(2)	1	4	3	2

ii	q[0]	q[1]	q[2]	q[3]
push(4)	4	0	0	0
push(3)	3	4	0	0
push(2)	2	4	3	0
push(1)	1	4	3	2

iii	q[0]	q[1]	q[2]	q[3]
push(4)	4	0	0	0
push(2)	2	4	0	0
push(3)	2	4	3	0
push(1)	1	4	3	2

Since the values of q are the same for all i, ii and iii after push, the output will also be the same. The output is 1234 for options i, ii and iii.

12 D Consider the truth tables below:

(A OR B) OR (A XOR B)		
	A = false	A = true
B = false	false	true
B = true	true	true
(A OR B) XOR (A XOR B)		
	A = false	A = true
B = false	false	false
B = true	false	true
(A OR B) OR (A AND B)		
	A = false	A = true
B = false	false	true
B = true	true	true
(A OR B) XOR (A NOR B)		
	A = false	A = true
B = false	true	true
B = true	true	true

A $\otimes$ B is equivalent to (A OR B) XOR (A NOR B).

13 D A pair of Boolean expression is logically equivalent if they have the same truth table. The truth tables are as follows:

`((NOT a) AND b) OR (a AND (NOT b))`		
	a = false	a = true
b = false	false	true
b = true	true	false
`NOT (a = b)`		
	a = false	a = true
b = false	false	true
b = true	true	false
`NOT ((NOT a) = (NOT b))`		
	a = false	a = true
b = false	false	true
b = true	true	false

From the truth tables, all three Boolean expressions are logically equivalent.

14 B Bun can always set the counter to a multiple of 8 after his round(regardless of Apple's choice). After reaching 992(multiple of 8), the counter must lie within 993-999 after Apple's round. Consequently, Bun can win the game.

15 A i. Bun can only use the strategy above if and only if the initial value of the counter is a multiple of 8, otherwise Apple can use the strategy above instead.

ii. Each time Apple used 0, Bun can use 0 in the next round to keep the counter to a multiple of 8.

iii. Using the strategy above, for each number `x` in 1 to 7, the count of number `x` used by Apple must be the same as the count of number `(8-x)` used by Bun after each of Bun's rounds, so Bun can always keep the counter to a multiple of 8.

So, the answer is i only.

16 D The program outputs the number of 1s in the binary notation of `x`. Since $79622_{10} = 10011011100000110_2$, the answer is 8.

17 B The program performs bubble sort on the odd index elements and even index elements of array `a` respectively. So only ii and iii must be true.

18 D

i. 65535 mod 3 = 0. Note that the range starts from 0, so the number of values % 3 that return 0 is 65535/3+1 = 21846. While that of 1 and 2 are 21845. The chance of returning 0 is higher than that of 1 and 2.

ii. (r()+r()+r()) mod 3 = (r() mod 3 + r() mod 3 + r() mod 3) mod 3.
From (i), we know that the probability function of r() mod 3 is not equally distributed. In fact, the probabilities of getting 0, 1, 2 are as follows,
P(0) = 21846 / 65535, P(1) = 21845 / 65535, P(2) = 21845 / 65535

For (r() mod 3 + r() mod 3 + r() mod 3) mod 3, the probabilities of getting 0, 1, 2 can be obtained by using the results above.

- P'(0) = P(0)P(0)P(0) + P(0)P(1)P(2) + P(0)P(2)P(1) + P(1)P(0)P(2) + P(1)P(1)P(1) + P(1)P(2)P(0) + P(2)P(0)P(1) + P(2)P(1)P(0) + P(2)P(2)P(2)
- P'(1) = P(0)P(0)P(1) + P(0)P(1)P(0) + P(0)P(2)P(2) + P(1)P(0)P(0) + P(1)P(1)P(2) + P(1)P(2)P(1) +P(2)P(0)P(2) + P(2)P(1)P(1) + P(2)P(2)P(0)
- P'(2) = P(0)P(0)P(2) + P(0)P(2)P(0) + P(0)P(1)P(1) + P(1)P(0)P(1)+ P(1)P(1)P(0) + P(1)P(2)P(2) + P(2)P(0)P(0) + P(2)P(1)P(2) + P(2)P(2)P(1)

Suppose P(0) = a, P(1) = P(2) = b,
$P'(0) = a^3 + 6ab^2 + 2b^3$, $P'(1) = P'(2) = 3a^2b + 3ab^2 + 3b^3$,
P'(0) > P'(1) = P'(2)
So, the chance of returning 0 is higher than that of 1 and 2.
Alternatively, considering r() that return an integer between 0 and 4 inclusively with equal probability would provide insights for finding that the chance of returning 0 is higher.

19 D

The possible range of (myrand(50) - 30) is [-30,19]. But after (mod 5), the range will become [-4,4], So the answer is 9.

20 B

Values of `i, x, y` after the `i`th iteration:

i	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
x	0	4	4	2	2	3	3	1	1	**5**
y	0	0	1	1	0	0	3	3	1	1

21 C

The push function pushes an element into the queue. The pop function outputs the first element in the queue and pops it. The queue size is 3. After the first 3 push, tail = head so the first pop outputs "Empty". Queue is a First-In-First-Out data structure, so the remaining outputs are "4", "8", "Empty".

22 C Calculate the number of different paths for every cell.

	1	2	3	4	5	6	7	8
1	1(A)	1	1	1	1	1	1	1
2	1	2	2	2	2	2	2	1
3	1	2	4	4	4	4	2	1
4	1	2	4	8	8	4	2	1
5	1	2	4	8	16	20	2	1
6	1	2	4	4	20	40	42	1
7	1	2	2	2	2	42	84	85
8	1	1	1	1	1	1	85	170(B)

There are 170 different paths.

23 A When `a[i]≠0`, `x=a[i]`, so `a[j] mod x` must be 0 when `j=i`. This sets `flag` to true and increases `res`. By tracing the program, it can be found that these values are not set to 0:

`a[0]` = 2, which sets 6(`a[2]`), 18(`a[5]`) and 50(`a[9]`) to 0

`a[1]` = 5, which sets 15(`a[4]`), 35(`a[7]`), and 45(`a[8]`) to 0

`a[3]` = 9

`a[6]` = 21

Alternatively, one may observe that the program outputs the number of elements in `a` that is not a multiple of any element before it. Only `2,5,9,21` meet this criterion, so `res = 4`.

24 B The possible scores of each round:

Round	Score
0	0:0
1	1:0 or 0:1
2	2:0 or 0:2
3	2:1 or 1:2 or 3:0 or 0:3
4	3:1 or 1:3
5	3:2 or 2:3
6	3:3

For round 1 and round 3, the probability of getting the score listed on the table from the previous round is 1. For the remaining rounds, the probability is 1 / 2. So, the final answer is $(1 / 2)^4 = 1 / 16$.

25 D The expected time require for each strategy:

	Expected Time
Strategy A	10*0.1 + 20*0.2 + 35*0.2 + 75*0.5 = 49.5
Strategy B	40*0.5 + 55*0.2 + 65*0.2 + 75*0.1 = 51.5
Strategy C	10*0.1 + 50*0.5 + 65*0.2 + 75*0.2 = 54
Strategy D	10*0.2 + 25*0.2 + 65*0.5 + 75*0.1 = 47

As strategy D has the minimum expected time among all strategies, it is the answer.

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>(s[n]='0')and((n=1)or(s[n-1]='0'))</td><td colspan="2">s[n-1]=='0'&&(n==1||s[n-2]=='0')</td></tr>
<tr><td colspan="3">Please be noted that 0 is also a multiple of 100. So, checking if the integer is 0 or the last two digits of the integer are both 0 would be correct.</td></tr>
<tr><td rowspan="2">B</td><td>b[i]:=b[i]+b[i-1]</td><td>b[i]=b[i]+b[i-1]</td><td>b[i]=b[i]+b[i-1]</td></tr>
<tr><td colspan="3">Observe that b[1] = a[1], b[2] = a[2] - a[1], b[3] = a[3] - a[2]...
So adding b[i-1] to each b[i] from i = 1 to i = 8 will turn every b[i] to be a[i] again.</td></tr>
<tr><td rowspan="2">C</td><td>b[9-i]:=b[9-i]-b[8-i]</td><td>b[9-i]=b[9-i]-b[8-i]</td><td>b[9-i]=b[9-i]-b[8-i]</td></tr>
<tr><td colspan="3">As b[i] stores the value from a[1] to a[i], b[i] - b[i-1] would be a[i].
But starting the process from b[1] would not work as b[i-1] in b[i] - b[i-1] has been modified for other i.
Reversing the order of the process by starting from b[8] could prevent the problem.</td></tr>
<tr><td>D1</td><td colspan="3" rowspan="4">This question is cancelled.</td></tr>
<tr><td>D2</td></tr>
<tr><td>D3</td></tr>
<tr><td>E</td></tr>
<tr><td rowspan="2">F</td><td>4</td><td>4</td><td>4</td></tr>
<tr><td colspan="3">f(x) returns the number of factors x. The factors of 10 are 1, 2, 5 and 10, there are a total of 4 factors.</td></tr>
<tr><td rowspan="2">G</td><td>3</td><td>3</td><td>3</td></tr>
<tr><td colspan="3">The factors of 121 are 1,11 and 121, there are a total of 3 factors.</td></tr>
<tr><td rowspan="2">H</td><td>25</td><td>25</td><td>25</td></tr>
<tr><td colspan="3">The program outputs the number of integers between 1 and 10000 having 3 factors. The integers having 3 factors must be square of prime numbers, so the program outputs the number of prime numbers between 1 and 100, which is 25.</td></tr>
<tr><td rowspan="2">I</td><td>11</td><td>11</td><td>11</td></tr>
<tr><td colspan="3">The xor(Pascal)/^(C, C++) function carries out element-wise xor operation on the binary representations of the two numbers. (e.g. 1100_2 xor 0101_2=1001_2)
By tracing the program, the result is 1011_2, which is 11_{10}</td></tr>
</table>

<table>
<tr><td>J1</td><td>start+1</td><td>start+1</td><td>start+1</td></tr>
<tr><td rowspan="2">J2</td><td>acc xor i</td><td>acc^i</td><td>acc^i</td></tr>
<tr><td colspan="3">By tracing g(n), the following is observed:
when n mod 4 = 0, g(n) = n
when n mod 4 = 1, g(n) = 1
when n mod 4 = 2, g(n) = n+1
when n mod 4 = 3, g(n) = 0
The main idea of f(n) is to reduce the number of operations using the observation above. J2 should be (acc xor i) as start+1 to n are the remaining parts of the calculation.
In f(n), start equals the largest number smaller than or equal to n that mod 4 = 2. From the observation above, in order to match the answer, acc here should have the value start+1, which can also correctly calculate the answers for other cases (when i = start+1, acc xor i = (start+1) xor (start+1) = 0, etc.).</td></tr>
<tr><td rowspan="2">K</td><td>f(a-m)+f(b-m)+f(c-m)</td><td>f(a-m)+f(b-m)+f(c-m)</td><td>f(a-m)+f(b-m)+f(c-m)</td></tr>
<tr><td colspan="3">f(x) returns the absolute value of x. f(x-m) calculates the difference between x and m. One of f(a-m), f(b-m), f(c-m) is equal to 0 and the other two’s sum will be equal to the range.</td></tr>
<tr><td rowspan="2">L</td><td>-480</td><td>-480</td><td>-480</td></tr>
<tr><td colspan="3">The error of the program is that it would treat the “+” sign as a digit precedent to the next number.
Given that you didn’t memorize the ASCII code of “+”, the digit that it represents can be figured out from the given example 100 + 1 = 51. Suppose x is the value “+” represents, solving 100 + 10x + 1 = 51, we will get x = -5.
The given expression 10 + 10 would evaluate to be 10 + -5 * 100 + 10 = -480</td></tr>
<tr><td>M1</td><td>26</td><td>56</td><td>85</td></tr>
<tr><td rowspan="2">M2</td><td>inc(i) end; //continue end;</td><td>i++; } // continue; }</td><td>i++; } // continue; }</td></tr>
<tr><td colspan="3">To fix the bug of the program, the program would simply need to go onto the next iteration whenever a “+” sign is met. This can be done by increasing the pointer (which is i) by 1, or using continue to break out of the current iteration and continue with the next.</td></tr>
</table>