Hong Kong Olympiad in Informatics 2017/18
Heat Event (Junior Group)
Official Solution

Statistics (N = 256)
Full mark = 45. Maximum = 40. Median = 13. Advance to Final = 17.5 marks or above.

Section A

Q	A	Explanation
1	T	Both the size of a character (char) variable and that of a Boolean (Pascal: `boolean`, C/C++: `bool`) variable are 1 byte.
2	T	It is possible to write a program without using any IDE. One of the ways is to code using a text editor, then pass the text file to a compiler.
3	F	Array indexes of `a[]` can only be integers between 0 and 9.
4	F	The maximum number of swaps required is N-1. One example is [2, 3, 4, 5, 6, 1] which requires 5 swaps to sort the array ascendingly.
5	F	404 means that the client is able to communicate with a given server, but the server could not find what was requested. There is no internal server error.
6	C	Ways required: 10C1 (ways to choose 1 president from 10 members) * 9C2 (ways to choose 2 presidents from 9 members, after selecting the president) = 10 * 9 * 8 / 2 = 360.
7	A	ASCII supports English alphabets, numbers, symbols and non-printing characters but not Chinese characters.
8	C	Total number of possible answers = 16 - (-16) + 1 = 33 The optimal strategy is to eliminate half of the possible answers each time. The maximum number of questions needed to know the exact value of the integer = $\lceil \log_2 33 \rceil$.
9	D	Note that the question says "if Charlie gets full marks in the final exam, he feels happy". It is possible that Charlie feels happy even if he doesn't get full marks in the final exam. Same for "if Charlie feels happy and it is raining outside, he plays computer games at home".
10	C	It's impossible for "`a = b`" and "`a < b`" to happen at the same time. So option (i) is wrong. While it's impossible to have "`a > b`" because `x-1` must be smaller than `x+1`, hence `(x - 1) / 2` must not be larger than `(x + 1) / 2`.
11	B	`x div 10` (`x / 10` in C and C++) = 12 `x mod 10` (`x % 10` in C and C++) = 3 12 + 100 * 3 = 312

12	A	There are only 3 possible pairings: (1) W vs X; Y vs Z (2) W vs Y; X vs Z (3) W vs Z; X vs Y For pairing (1), W and Z will advance to the finals and Z always wins. Hence (i) is an impossible to happen. For pairing (2), Y and X will advance to the finals and Y always wins. Hence (ii) is possible to happen. For pairing (3), Z and Y will advance to the finals and Z always wins. Final match between Y and W is never impossible to happen, so (iii) is wrong.
13	D	(i): If `a = b`, the value of `a or b` (`a | b` in C and C++) and `a and b` will both be equal to a (and also b). When values of a and b differ, values of `a or b` and `a and b` will not be equal, hence (i) is correct. (ii): For xor operations, if two digits of the same position are the same, 0 will be returned. On the other hand, if two digits of the same position aren't the same, 1 will be returned. `a xor b` would be 0 if and only if a = b. (ii) is correct.
14	D	Value of `s[i + arr[i]]` is replaced by that of `s[i + 1]` in each for-loop. When `i` = 1, the (1 + 1) 2nd character is replaced by (1 + 1) 2nd character in `s`. `s` = "hkoi201718" When `i` = 2, the (2 + 4) 6th character is replaced by (2 + 1) 3rd character in `s`. `s` = "hkoi2o1718" When `i` = 3, the (3 + 0) 3rd character is replaced by (3 + 1) 4th character in `s`. `s` = "hkii2o1718" When `i` = 4, the (4 + 3) 7th character is replaced by (4 + 1) 5th character in `s`. `s` = "hkii2o2718" When `i` = 5, the (5 + 5) 10th character is replaced by (5 + 1) 6th character in `s`. `s` = "hkii2o271o" When `i` = 6, the (6 + 2) 8th character is replaced by (6 + 1) 7th character in `s`. `s` = "hkii2o221o"
15	B	A: output = 36 B: output = 51 * C: output = 40 D: output = 25
16	C	Note that in the end, only when `i` is even, `a[i]` would be equal to `i`. Hence the sum = 20 + 18 + 16 + … + 4 +2 = (2 + 20) * 10 / 2 = 110
17	B	A: output = 28 32 B: output = 30 30 * C: output = 31 29 D: output = 32 28

18	C	The program returns the digits sum of `x`. Digits sum of `x` = 2 + 0 + 1 +8 = 11
19	C	`f` is implementing the push operation of a stack while `g` is implementing the pop operation of a stack.
20	D	Value of `x` and `y` that results in output of "ab" can be 9 and 10. Value of `x` and `y` that results in output of "ac" can be 5 and 6. Value of `x` and `y` that results in output of "bc" can be 11 and 10.
21	D	(i) cannot be re-ordered to form a palindrome as frequencies 'A' and 'C' are odd. (ii) can be re-ordered to form XYZZZYX, which is a palindrome. (iii) can be re-ordered to form PPQRSSSSRQPP, which is a palindrome.
22	D	(i) cannot be re-ordered to form a palindrome as frequencies of 'A', 'B' and 'C' are different. (ii) can be re-ordered to form HKOIHKOI, which is a periodic string. (iii) can be re-ordered to form IOIPPPIOIPPP, which is a periodic string.
23	A	`f(n)` returns the value with the trailing 0s of `n` in binary representation removed. Return value of f(65) = 65 ($65_{10} = 1000001_2$) * Return value of f(122) = 61 ($122_{10} = 1111010_2$, $61_{10} = 111101_2$) Return value of f(4032) = 63 ($4032_{10} = 111111000000_2$, $63_{10} = 111111_2$) Return value of f(65536) =1 ($65536_{10} = 10000000000000000_2$)
24	A	We just need to access `a[499]` to obtain the value of `x`.
25	A	Notice that A nor B $\equiv$ (not A) and (not B). Option A: A nor (B nor B) $\equiv$ (not A) and (not((not B) and (not B))) $\equiv$ (not A) and (not(not B)) $\equiv$ (not A) and B *

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>A1</td><td>a+x</td><td>a+x</td><td>a+x</td></tr>
<tr><td rowspan="2">A2</td><td>5050-a</td><td>5050-a</td><td>5050-a</td></tr>
<tr><td colspan="3">By subtracting the sum of a from the sum of adding up 1 to 100 [(1 + 100) * 100 / 2 = 5050], the target number can be found.</td></tr>
<tr><td>B</td><td>t:=a[x];a[x]:=a[y];
a[y]:=t</td><td>t=a[x];a[x]=a[y];
a[y]=t</td><td>t=a[x];a[x]=a[y];
a[y]=t</td></tr>
<tr><td rowspan="2">C</td><td>k-i-1 // k-i // k div 2 //
(k+1) div 2</td><td>k-i-1 // k-i // k/2 //
(k+1)/2</td><td>k-i-1 // k-i // k/2 //
(k+1)/2</td></tr>
<tr><td colspan="3">To ensure that the swapping isn’t done to the same pair of characters twice.</td></tr>
<tr><td rowspan="2">D</td><td>f(100);f(k);f(100)</td><td>f(100);f(k);f(100)</td><td>f(100);f(k);f(100)</td></tr>
<tr><td colspan="3">The first f(100) is to reverse the whole array. f(k) is to reverse the last k elements in the original array in the original order. Then f(100) follows is to reverse the whole array back again. In the end only the last k elements are reversed.</td></tr>
<tr><td>E</td><td colspan="3">E1 A5</td></tr>
<tr><td rowspan="2">F</td><td colspan="3">E1 C5 / E1 A3 / E3 A5 / C1 A5</td></tr>
<tr><td colspan="3">First of all, we need to know the number of valid paths passing through each cell in the grid. To calculate this, we can first find out the number of ways a path from S to T can reach each cell (dp1) and the number of ways a path from T to S can reach each cell (dp2) (which is equal to dp1 in this case as the grid is a square). For a cell in row (R+1) and row (C+1), the number of ways = (R+C)! / (R! C!). We can then multiply dp1[i][j] *dp2[i][j] for each cell (i, j) to get the information.
Back to the problem, we can notice that if two Cs are placed with a “top right, bottom left” relation, there won’t be a path passing through both Cs. If we place the two Cs like this, the number of interesting paths would be the sum of number of paths passing through these two cells. (Of course we can place the Cs in other ways, but the problem will get more complicated as we need to subtract the number of paths passing through more than 1 C) Hence we can just choose two cells with their path sum = the number required by the question.</td></tr>
<tr><td rowspan="2">G</td><td>96</td><td>96</td><td>96</td></tr>
<tr><td colspan="3">The worst case can be “abcdefg…xyzabcdefg…xyz…”. In this case, we can keep all of any one of the most frequently appeared letter and the frequency would be ⌈100/26⌉ = 4. Hence 100 – 4 = 96 characters need to be removed in the worst case.</td></tr>
</table>

H	`x xor 32`	`x^32`	`x^32`
	Notice that the ASCII value of any lowercase letter and its corresponding uppercase letter differs by 32 (For example 'A' = 65; 'a' = 97). 32 is a power of two, which is exactly a digit value of a binary number. By doing `x xor 32`, the 6th bit of the ASCII value (`tmp`) is converted to 1 if it is 0, or it's converted to 0 if it's 1 originally, meaning that a subtraction/ addition of 32 to the ASCII value (`tmp`) is done.		
I1	`num[i]+abs(num[i])`	`num[i]+abs(num[i])`	`num[i]+abs(num[i])`
I2	`temp div 2`	`temp/2`	`temp/2`
	When `num[i]` is positive, `num[i]+abs(num[i])` will add the value of `num[i]` to `temp` twice. When `num[i]` is negative, `num[i]` and `abs(num[i])` will actually eliminate each other, ending up with nothing added to `temp`. Since for each of the positive values `num[i]` is added to temp twice, dividing `temp` by two will lead to a correct answer.		
J1	`i:=2;`	`i=2;`	`i=2;`
	2 is the smallest possible factor of `n`. (excluding 1)		
J2	`n mod i=0`	`n%i==0`	`n%i==0`
	There exist a factor other than 1 and `n`.		
K	`answer+is_prime(i)`	`answer+is_prime(i)`	`answer+is_prime(i)`
L1	`29`	`58`	`88`
L2	`for i:=0 to 10000 do`	`for (i=0;i<=10000;i++) // for(i=0;i<10001;i++)`	
	This for-loop should be iterating the range of input (i.e. 0 to 10000) but not the number of integers inputted.		