HKOI 2017/18 Final Event Briefing Session

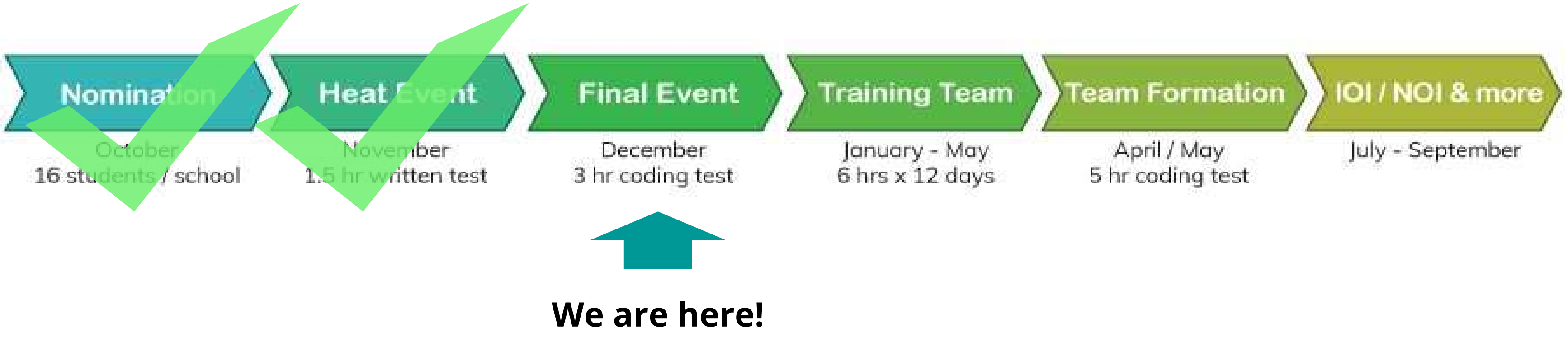
LAU Chi Yung, Steven
CHOW Kwan Ting Jeremy
HO Ngan Hang, Anson

香港電腦奧林匹克競賽
Hong Kong Olympiad in Informatics

Content

1. Rules and Procedure
2. Identify common mistakes & sharing session
3. Practice Competition
4. Solution and demonstration

HKOI Timeline



Benefits of winning HKOI

1. Get a medal

- ~180 contestants in total, ~90 contestants in each group
- ~Top 50% of contestants in each group will get a medal
- Gold : Silver : Bronze ≈ 1 : 2 : 3

Benefits of winning HKOI

2. Enter the HKOI Training Team

- Trainings on every Saturday from Feburary to May
- Lectures, coding practices, mini-competitions, social events...
- Meet friends!

HKOI BBQ 2016/17

Benefits of winning HKOI

3. Be eligible to join the Team Formation Test (TFT)

TFT selects potential students to participate in:

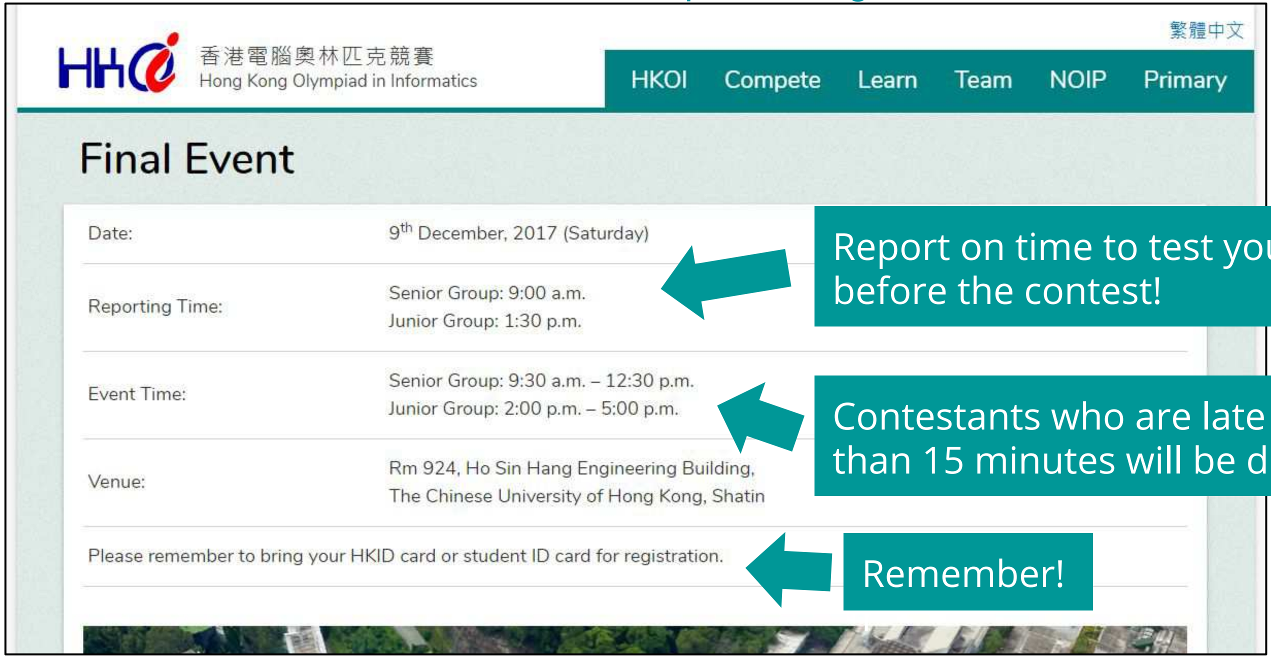
- International Olympiad in Informatics (IOI)
  - Hosts: **Japan (2018)**, Iran (2017), Russia (2016), Kazakhstan (2015)
- National Olympiad in Informatics (NOI)
  - Shaoxing 紹興 (2017), Mianyang 綿陽 (2016), Hangzhou 杭州 (2015)
- ACM-ICPC Hong Kong Chapter

HKOI 2017/18 Final Event

Contest Environment

Final Event

http://hkoi.org/en/final-event-2017-18



Final Event

Date:	9th December, 2017 (Saturday)
Reporting Time:	Senior Group: 9:00 a.m. Junior Group: 1:30 p.m.
Event Time:	Senior Group: 9:30 a.m. – 12:30 p.m. Junior Group: 2:00 p.m. – 5:00 p.m.
Venue:	Rm 924, Ho Sin Hang Engineering Building, The Chinese University of Hong Kong, Shatin

Please remember to bring your HKID card or student ID card for registration.

Report on time to test your machine before the contest!

Contestants who are late for more than 15 minutes will be disqualified

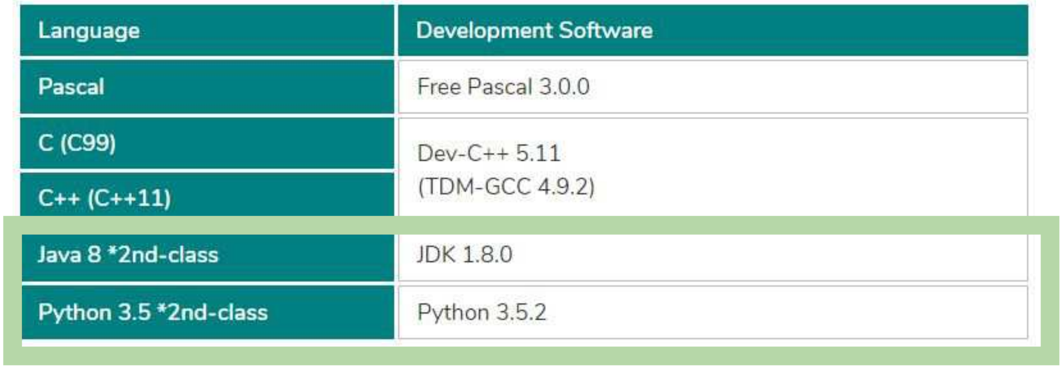
Remember!

Final Event

Programming languages

- We cannot guarantee that the problems are solvable using **Java** and **Python**
- We cannot guarantee the proper functioning of the software provided for **Java** and **Python**
- Contestants may use such languages at their own risk

http://hkoi.org/en/rules-2017-18



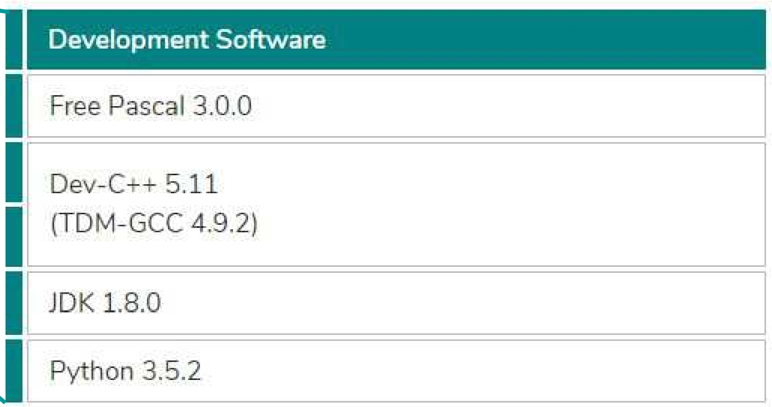
Language	Development Software
Pascal	Free Pascal 3.0.0
C (C99)	Dev-C++ 5.11 (TDM-GCC 4.9.2)
C++ (C++11)	
Java 8 *2nd-class	JDK 1.8.0
Python 3.5 *2nd-class	Python 3.5.2

Final Event

Software

- Desktop Computer (Windows 7)
- Visual Studio Code
  (with Pascal, C/C++, Java and Python plugins)
- You can use any software provided
  (paint, calc, IDE, compiler, web browser etc)
- C++ and Pascal documentations will be provided in the web browser
- NO Internet except HKOI Online Judge

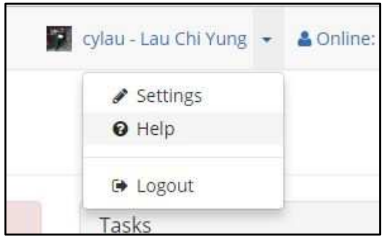
However, submitted programs will be compiled under the Linux operating system



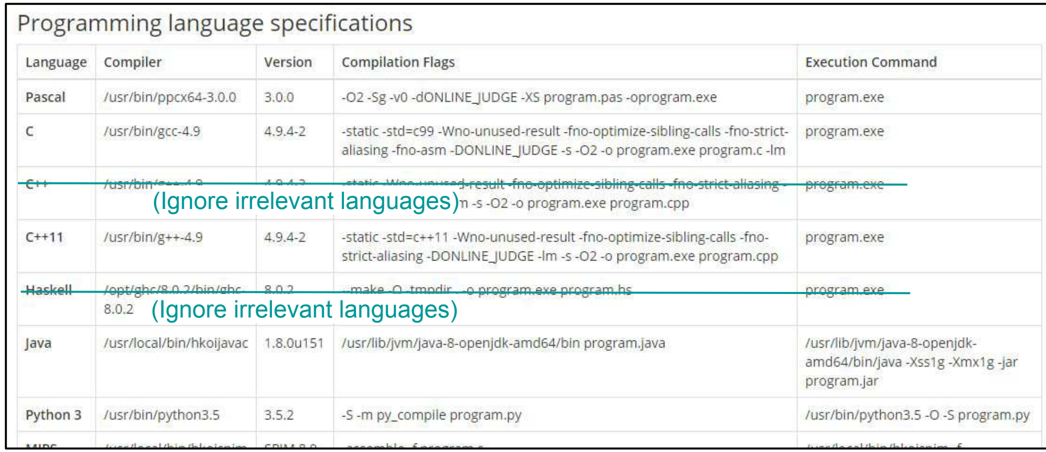
Development Software
Free Pascal 3.0.0
Dev-C++ 5.11 (TDM-GCC 4.9.2)
JDK 1.8.0
Python 3.5.2

香港電腦奧林匹克競賽
Hong Kong Olympiad in Informatics

Final Event



https://judge.hkoi.org/help



Programming language specifications

Language	Compiler	Version	Compilation Flags	Execution Command
Pascal	/usr/bin/ppcx64-3.0.0	3.0.0	-O2 -Sg -v0 -dONLINE_JUDGE -XS program.pas -oprogram.exe	program.exe
C	/usr/bin/gcc-4.9	4.9.4-2	-static -std=c99 -Wno-unused-result -fno-optimize-sibling-calls -fno-strict-aliasing -fno-asm -DONLINE_JUDGE -s -O2 -o program.exe program.c -lm	program.exe
~~C++~~	~~/usr/bin/g++-4.9~~	~~4.9.4-2~~	~~-static -Wno-unused-result -fno-optimize-sibling-calls -fno-strict-aliasing -~~ [illegible]m -s -O2 -o program.exe program.cpp	~~program.exe~~
C++11	/usr/bin/g++-4.9	4.9.4-2	-static -std=c++11 -Wno-unused-result -fno-optimize-sibling-calls -fno-strict-aliasing -DONLINE_JUDGE -lm -s -O2 -o program.exe program.cpp	program.exe
~~Haskell~~	~~/opt/ghc/8.0.2/bin/ghc-~~ 8.0.2	~~8.0.2~~	~~--make -O -tmpdir . -o program.exe program.hs~~	~~program.exe~~
Java	/usr/local/bin/hkoijavac	1.8.0u151	/usr/lib/jvm/java-8-openjdk-amd64/bin program.java	/usr/lib/jvm/java-8-openjdk-amd64/bin/java -Xss1g -Xmx1g -jar program.jar
Python 3	/usr/bin/python3.5	3.5.2	-S -m py_compile program.py	/usr/bin/python3.5 -O -S program.py
MIPS	/usr/local/bin/hkoisim	SPIM 8.0	-assemble -f program.s	/usr/local/bin/hkoisim -f

(Ignore irrelevant languages)

(Ignore irrelevant languages)

Software

- You can view the compiler flags on the HKOI Online Judge, even during contest
- You will develop your solutions on Windows 7
- Submitted programs will be compiled under the Linux operating system
- There might be differences in compiler behaviour between Windows and Linux in rare occasions
- We will not help resolve errors related to this during contest
- Please test it using your HKOI Online Judge account in this week to avoid using strange syntax

香港電腦奧林匹克競賽
Hong Kong Olympiad in Informatics

Final Event

Hardware

- Roughwork sheet, keyboard, mouse and mousepad will be provided
- You can bring one personal keyboard for use in the competition
  - Wireless keyboards, keyboards that require installation of drivers, and mechanical keyboards fitted with “blue” switches (or equivalent) are not allowed
  - We reserve the right to examine and disallow any keyboard.
- Your own stationery (pen, pencil, rubber, ruler etc)
- NO calculators or other electronic devices
- NO personal roughwork sheet

HKOI 2017/18 Final Event

Question Paper

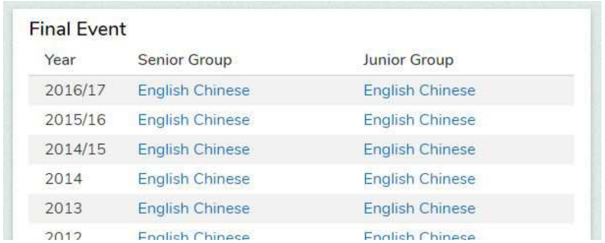
香港電腦奧林匹克競賽
Hong Kong Olympiad in Informatics

Final Event

Question paper

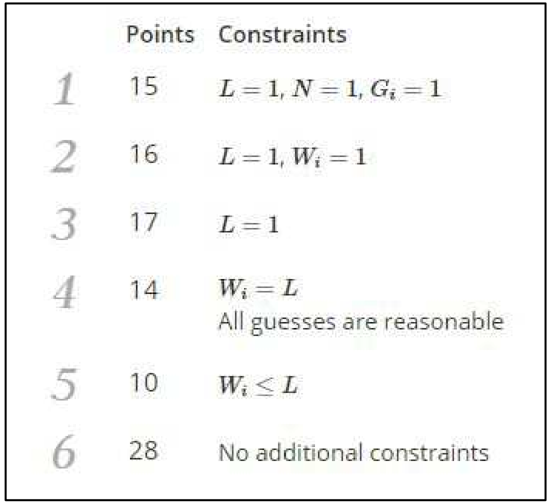
- There are four tasks in total
- Each task worths 100 points
- Each task is divided into subtasks with different constraints and points

http://hkoi.org/en/past-problems



Final Event

Year	Senior Group	Junior Group
2016/17	English Chinese	English Chinese
2015/16	English Chinese	English Chinese
2014/15	English Chinese	English Chinese
2014	English Chinese	English Chinese
2013	English Chinese	English Chinese
2012	English Chinese	English Chinese



	Points	Constraints
1	15	$L = 1, N = 1, G_i = 1$
2	16	$L = 1, W_i = 1$
3	17	$L = 1$
4	14	$W_i = L$ All guesses are reasonable
5	10	$W_i \le L$
6	28	No additional constraints

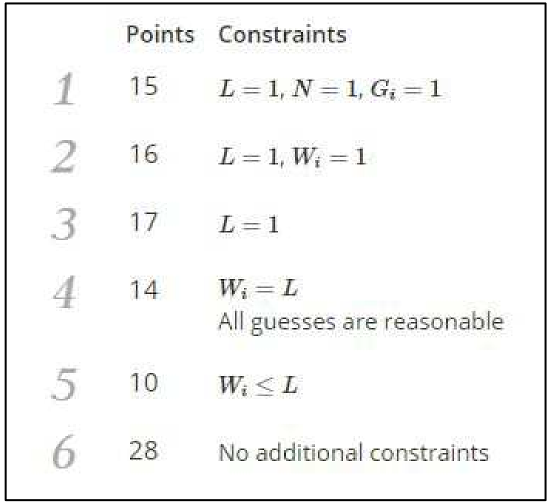
HKOI 2016/17 Final Event Junior Task 1 "Acronym"
https://judge.hkoi.org/task/J171

香港電腦奧林匹克競賽
Hong Kong Olympiad in Informatics

Final Event

Scoring

- If your solution passes ALL testcases in a subtask, you get all points of that subtask (a.k.a. Batch Scoring)
- For example, a solution solving all cases with L = 1 would get 15 + 16 + 17 = 48 points



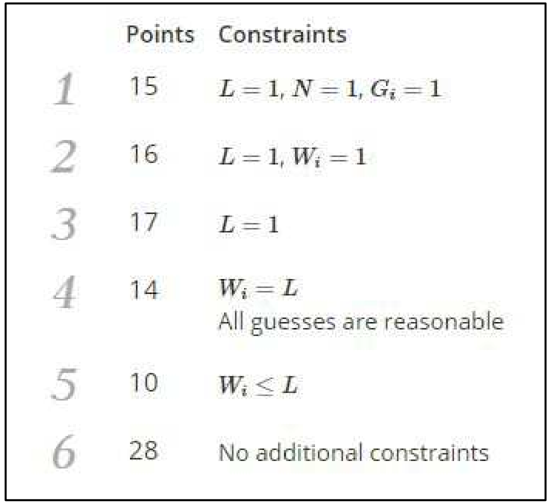
	Points	Constraints
1	15	$L = 1, N = 1, G_i = 1$
2	16	$L = 1, W_i = 1$
3	17	$L = 1$
4	14	$W_i = L$ All guesses are reasonable
5	10	$W_i \le L$
6	28	No additional constraints

HKOI 2016/17 Final Event
Junior Task 1 "Acronym"

Final Event

Scoring

- Scores of each subtasks are accumulated
- So, if you submit a solution that passes only subtask 1, you get 15 points; if you then submit another solution that passes only subask 2, your final score will be 15 + 16 = 31 points



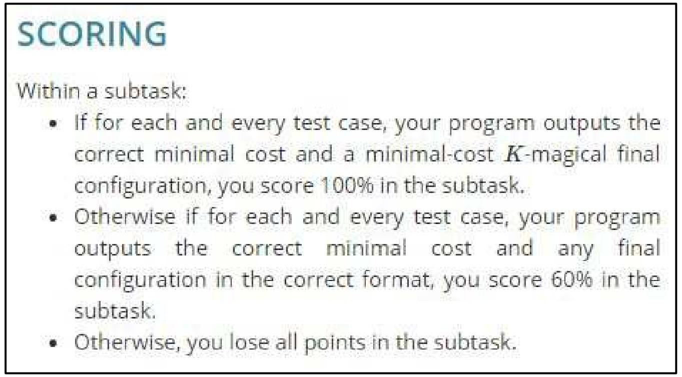
	Points	Constraints
1	15	$L = 1, N = 1, G_i = 1$
2	16	$L = 1, W_i = 1$
3	17	$L = 1$
4	14	$W_i = L$ All guesses are reasonable
5	10	$W_i \leq L$
6	28	No additional constraints

HKOI 2016/17 Final Event
Junior Task 1 "Acronym"

Final Event

Scoring

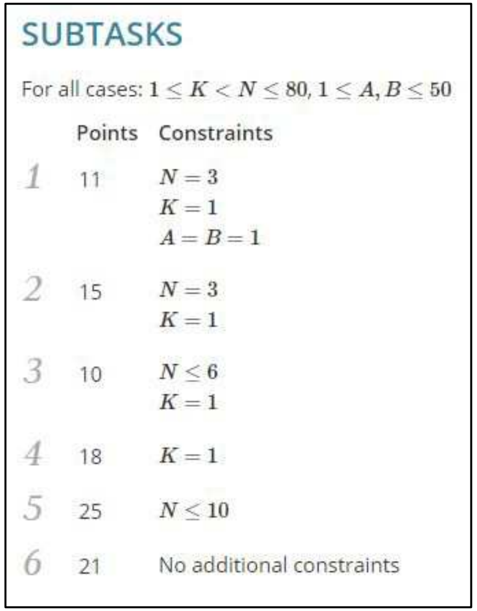
- Some tasks could employ partial scoring
- One possible score:
  60% * 11 + 100% * 15 = 21.6 points



SCORING

Within a subtask:

- If for each and every test case, your program outputs the correct minimal cost and a minimal-cost K-magical final configuration, you score 100% in the subtask.
- Otherwise if for each and every test case, your program outputs the correct minimal cost and any final configuration in the correct format, you score 60% in the subtask.
- Otherwise, you lose all points in the subtask.



SUBTASKS

For all cases: $1 \le K < N \le 80, 1 \le A, B \le 50$

	Points	Constraints
1	11	$N = 3$ $K = 1$ $A = B = 1$
2	15	$N = 3$ $K = 1$
3	10	$N \le 6$ $K = 1$
4	18	$K = 1$
5	25	$N \le 10$
6	21	No additional constraints

HKOI 2016/17 Final Event
Senior Task 1 "Magic Triangle II"

香港電腦奧林匹克競賽
Hong Kong Olympiad in Informatics

Final Event

Writing a solution

- Use standard input and standard output, not file I/O
- i.e. scanf, printf, cin, cout, read, readln, write, writeln
- avoid fopen, system("pause") etc
- For C/C++, main function should return 0
- Please make use of your HKOI Online Judge account to practice and test

We will demonstrate to you later

Final Event

Submitting solution

- Same procedure as in HKOI Online Judge
- You will receive feedback about your submission:
  the type of error first encountered (if any) for each subtask
- You may submit at most once per task per 60 seconds, and at most 50 times per task

We will demonstrate to you later

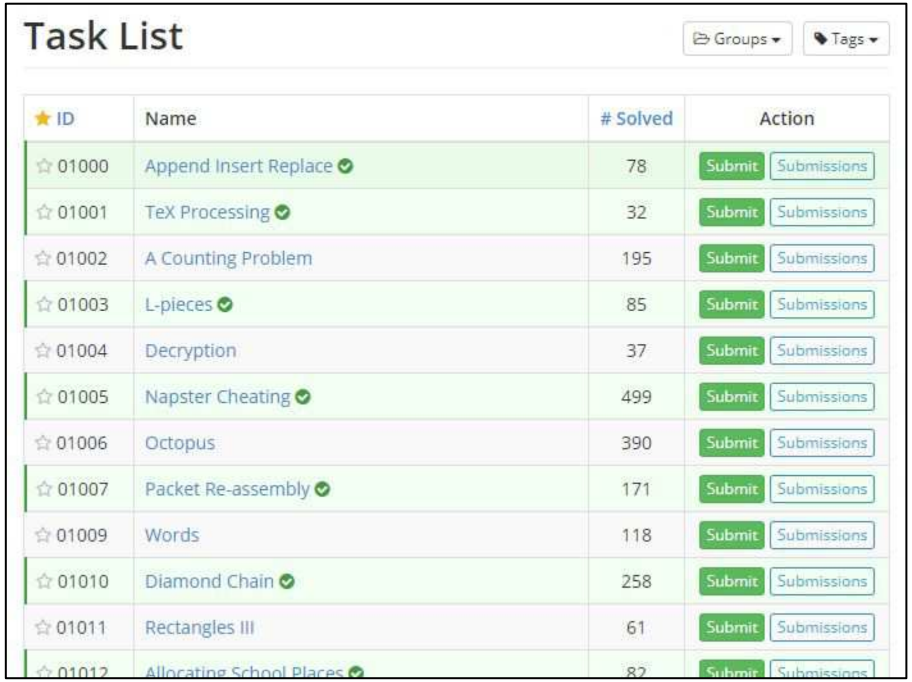
HKOI 2017/18 Final Event

One week to go, what should I do now?

Final Event

Practice!

- HKOI Online Judge http://judge.hkoi.org/
- Many tasks and virtual contests for practice
- Each finalist has been given a practice account
- Please make good use of it
- You may practice until 2017-12-09 00:00am (you are advised to sleep earlier!)



Task List

ID	Name	# Solved	Action
01000	Append Insert Replace	78	Submit Submissions
01001	TeX Processing	32	Submit Submissions
01002	A Counting Problem	195	Submit Submissions
01003	L-pieces	85	Submit Submissions
01004	Decryption	37	Submit Submissions
01005	Napster Cheating	499	Submit Submissions
01006	Octopus	390	Submit Submissions
01007	Packet Re-assembly	171	Submit Submissions
01009	Words	118	Submit Submissions
01010	Diamond Chain	258	Submit Submissions
01011	Rectangles III	61	Submit Submissions
01012	Allocating School Places	82	Submit Submissions

香港電腦奧林匹克競賽
Hong Kong Olympiad in Informatics

Final reminder

- Before leaving home, check your bag:
  - HKID or student ID
  - Pen, pencil, rubber, ruler
- Report on time

BRIEFING FOR HKOI 2017/18 FINALIST

2017-12-02

USEFUL TECHNIQUES

ONE WEEK TO PRACTICE

USEFUL TECHNIQUES

- Some simple algorithm / skills
  - Linear search
  - Binary search
  - Depth-first-search (DFS) / Breadth-first-search (BFS)
  - O(N^2) sort / Counting Sort
  - Partial Sum
- Simple mathematics
  - Pythagoras’s theorem
  - Finding primes / factors

USEFUL TECHNIQUES

- Some simple data structures
  - Queue / Stack / Linked list
- Data handling
  - Main tested skill in Junior
  - Basic skill for Senior
  - E.g. Time, date, string, array processing

USEFUL TECHNIQUES

- Exhaustion / Brute Force
  - Trying all possible cases
  - Good approach to some problems
  - Can be done with iteration (for loop / while) or recursion
- Time complexity evaluation
  - Estimate whether your algorithm can run within time limit
  - ~ 2 * 10^7 operations can be run in 1s
  - Optimize your algorithm if your time complexity is too high

PREPARATION BEFORE CONTEST

PRACTICE MAKE PERFECT

PREPARATION BEFORE CONTEST

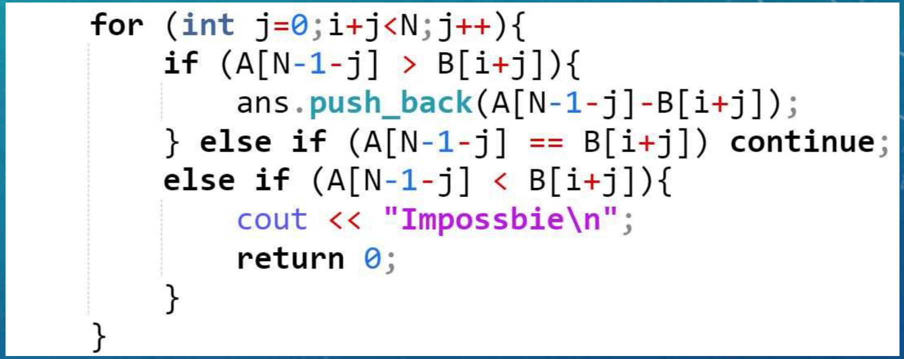
- Get familiar with coding
- Solve past papers of HKOI / other problems on HKOJ
- Practice on other programming site
  - E.g. Codeforces, Hackerrank
- Revision on basic algorithm
  - E.g. Sorting, binary search
- Revision on usage of some function
  - Lower_bound, strcpy (C / C++)
  - Copy, Length (Pascal)

COMMON MISTAKES

AVOID MAKING THOSE MISTAKES

COMMON MISTAKES

- The spelling and cases of output
  - “yes”, “Yes”, “Impossibie”, “TURE”
- Use correct datatype
  - E.g. don’t use integer to store decimal number
  - Sometimes the value of output maybe large -> overflow
  - Choice between signed 32-bit integer and 64-bit integer
  - longint(PASCAL), int(C++/C) / int64(PASCAL), long long(C++/C)
  - %lld instead of %d for (C++/C)



```
for (int j=0;i+j<N;j++){
    if (A[N-1-j] > B[i+j]){
        ans.push_back(A[N-1-j]-B[i+j]);
    } else if (A[N-1-j] == B[i+j]) continue;
    else if (A[N-1-j] < B[i+j]){
        cout << "Impossbie\n";
        return 0;
    }
}
```

COMMON MISTAKES

- Array size
  - Make sure you assign enough size for the array
  - Avoid negative index in C / C++
- Initialization
- Avoid doing useless things
  - Naïve hard coding
  - Small constant optimization
  - Randomize
  - Over complicated algorithm

COMMON MISTAKES

- Corner case, Boundary case
- Wrong time management
  - Waste too much time on a single task
  - Waste too much time on aiming full solution
  - Ignore some simple subtasks

STRATEGIES

WHAT SHOULD YOU DO

STRATEGIES – BEFORE CONTEST

- Relax
- Check the equipment (mouse / keyboard) carefully
- Check the programming environment carefully
  - E.g. Compile successfully ? Path of executable ?
- Try writing some simple program
  - E.g. Hello World, tasks in practice session

STRATEGIES – DURING CONTEST

- Read all problems before you start coding
  - Problems are not sorted by difficulty
  - Be patient to long problem statement
- Start with the problem you are most confident in
- Don’t always aim for full solution
  - Subtasks give you good amount of score
  - Some easy subtasks only need few lines of code
  - Sometime subtasks are hints, guide you to full solution

STRATEGIES – DURING CONTEST

- Be careful with the constraints
  - Some special constraints are hints
  - E.g. Distinct integer, maximum value of integer <= 100
- Don’t get panicked when your solution are not getting accepted
  - Correctness of your algorithm?
  - Corner / Boundary case?
  - Integer Overflow?
  - Size of Array is not large enough?
  - Divide by 0?

STRATEGIES – DURING CONTEST

- When you receive TLE (Time limit Exceed)
- Infinite loop?
- Analysis what is the bottleneck of your solution
  - Optimize your algorithm
  - E.g. Binary search instead of linear search
  - Try different approach
- Don’t hesitate to give up on a problem
  - When you feel like you won’t able to get more marks
  - When you spend too much time
  - Most candidates CANNOT solve ALL problems
  - Most candidates CANNOT completely solve ONE problem

STRATEGIES – DURING CONTEST

- Try to observe some special property
  - You need some observations to solve most of the tasks in HKOI
  - Wrong attempt does not deduct your scores
  - You may write some programs base on your assumption and submit
  - Test the correctness of your assumption
  - Use the assumption to optimize your algorithm -> Full solution

- 2015/16 Senior “Military Training”
  - It is guaranteed that Robo's position at time K will not be (r0,c0).
  - It is sufficient to find the answer by simulating the move of Robo’s by K x K times instead of N x N times

STRATEGIES – DURING CONTEST

- Use good approach to debug
  - Don’t just sit there and think
  - Output the value of some variables and compare with your expected value
  - Check with samples and your own test cases
  - Use slow but accurate program to debug (Advance)
- Read the problem statement again to make sure you didn’t miss any parts