CSCI-UA 480.4: APS
Algorithmic Problem Solving

Introduction to the Class

Instructor: Joanna Klukowska

created based on materials for this class by
Bowen Yu and materials shared by the authors of
the textbook Steven and Felix Halim

This Course

- Course website: https://cs.nyu.edu/~joannakl/aps_s19/

  This page contains the syllabus and daily summaries as well as loads of links to all other resources and services you will need for this class.

- Recitations (required):
  - Fridays 5:10 - 7:00pm
  - plan to bring your laptop
- Course message board / discussion: Piazza
  - you can self-sign up at https://piazza.com/nyu/spring2019/aps
- Online judge: Vjudge
- Grades posted on NYU Classes
- (Possibly Gradescope - not certain yet)

Why are you here?

What are we going to do?

- Basic and advanced data structures and algorithms
    - review the ones you know (data structures, basic algoriths, *Cracking the Code Interview*, ...)
    - learn a few new ones
    - use them to solve interesting problems
    - decide what data structues and algorithms should be used for which problems
- Programming in Java and C/C++
    - practice your coding skills in both
    - practice your debugging skills
    - practice testing skills
- Bugs, edge cases, problems, ...
    - learn how to appreciate problems
    - use the problems your find to learn for the future
    - learn from each other

Challenge

Write a program that reads two integers from standard input, calculates their sum and prints the result to standard output.

- use either Java or C/C++
- write the entire program
- write legibly

Example input:

Example output:

Challenge continued

Exchange your code with someone sitting next to you. The code that you get should be in the progrmming language that you are familiar with (i.e., if you do not know C++ do not take code that was written in C++).

Challenge continued

Exchange your code with someone sitting next to you. The code that you get should be in the progrmming language that you are familiar with (i.e., if you do not know C++ do not take code that was written in C++).

Working with your partner's code, decide

1. would it compile, if it was typed exactly the way it is written
2. would it produce the correct result if the input values were :
    -
    -
    -
    -
    -
3. is there something interesting, noteworthy, strange, ... in the code that you are reading?

Typical problems

Problem statement:

- description of the problem
- description of constraints
- description of the given input format and expected output format
- sample input and corresponding output

Typical problems

Problem statement:

- description of the problem
- description of constraints
- description of the given input format and expected output format
- sample input and corresponding output

Three parts for the problem solution:

- read the input data (from standard input)
- calculate results
- output the results (to standard output)

Typical problems

Problem statement:

- description of the problem
- description of constraints
- description of the given input format and expected output format
- sample input and corresponding output

Three parts for the problem solution:

- read the input data (from standard input)
- calculate results
- output the results (to standard output)

Online Judge (black-box testing) ⚖️

- most of the problems will be graded by an online judge
- contains many hidden tests (correctness of results and format matter!)
- produces instant result (the solution either passed or failed the test)
- objective
- readability of the code does not matter 😢 (well, at least note for an OJ)

Workflow

- Read and **understand** the problem.

Workflow

- Read and **understand** the problem.
- Think about possible solutions and pick one

Workflow

- Read and **understand** the problem.
- Think about possible solutions and pick one
- Analyze the solutions's correctness and efficiency (if not correct or not efficient go back to the previous step)

Workflow

- Read and **understand** the problem.
- Think about possible solutions and pick one
- Analyze the solutions's correctness and efficiency (if not correct or not efficient go back to the previous step)
- Write the solution as working code

Workflow

- Read and **understand** the problem.
- Think about possible solutions and pick one
- Analyze the solutions's correctness and efficiency (if not correct or not efficient go back to the previous step)
- Write the solution as working code
- Test your solution using the sample input/output given in the problem

Workflow

- Read and **understand** the problem.
- Think about possible solutions and pick one
- Analyze the solutions's correctness and efficiency (if not correct or not efficient go back to the previous step)
- Write the solution as working code
- Test your solution using the sample input/output given in the problem
- Test your solution using your own tests (this means you need to create inputs for which you know the correct output)

Workflow

- Read and **understand** the problem.
- Think about possible solutions and pick one
- Analyze the solutions's correctness and efficiency (if not correct or not efficient go back to the previous step)
- Write the solution as working code
- Test your solution using the sample input/output given in the problem
- Test your solution using your own tests (this means you need to create inputs for which you know the correct output)
- Submit your solution to the online judge

Workflow

- Read and **understand** the problem.
- Think about possible solutions and pick one
- Analyze the solutions's correctness and efficiency (if not correct or not efficient go back to the previous step)
- Write the solution as working code
- Test your solution using the sample input/output given in the problem
- Test your solution using your own tests (this means you need to create inputs for which you know the correct output)
- Submit your solution to the online judge
- If it passes, celebrate! 💃

Workflow

- Read and **understand** the problem.
- Think about possible solutions and pick one
- Analyze the solutions's correctness and efficiency (if not correct or not efficient go back to the previous step)
- Write the solution as working code
- Test your solution using the sample input/output given in the problem
- Test your solution using your own tests (this means you need to create inputs for which you know the correct output)
- Submit your solution to the online judge
- If it passes, celebrate! 💃
- If it fails, go back to one of the previous steps depending on the reason for failure

Course Syllabus