Tutorial 01 - Basic C++, Basic OOP, Analysis

CS2040S Semester 1 2023/2024

Join the Telegram group

https://t.me/+O-nT1rW_em5jMjA1

Set real display name

https://pollev.com/rezwanarefin430

Hi! I am Rezwan Arefin

- Year 3 Computer Science student from Bangladesh.
- Never took CS1101S and CS2040S or equivalent before, learned the materials by doing competitive programming in high school.
- Second time teaching CS2040. Previously taught CS2040C.
- Quite active on Discord and Telegram @RezwanArefin01.

Self Introductions

- State:
  - Name
  - Year of Study
  - One interesting thing about yourself

Some Admin Info

- Announcements will be mailed to you **and/or** sent on Discord.
- Information about this module will be updated at: https://www.comp.nus.edu.sg/~stevenha/cs2040s.html
- Kattis automatically checks for plagiarism, with all previous submissions.
- There will be no recording for tutorials. Please do attend them if you can, since participation marks are given.
- Slides will be sent to Telegram after the session ends.

Some Admin Info

I will use Poll Everywhere to measure you activeness.
Only participation is expected; correctness is secondary.

The tutorial/lab participation marks returns to encourage class participation. These marks will be given by the tutor **at the end of the semester** using the following guideline:

- 0% if you only attend ≤ 5 out of 10 tutorial/lab sessions (we lose Monday of Week 13, due to Deepavali PH in-lieu),
- 1% for **at most the bottom three** most-passive students (assuming these students attend > 5 tutorial/lab sessions),
- 3% for **at least the top three** most-active students (answering questions when asked by TA – the correctness of your answers are secondary; or even just by asking your own questions to TA before/during/after class/during consultation); in each tutorial group, and
- 2% for the rest.

Agenda

Each session comprises of:

- **Tutorial:** Go through the questions and clarify any doubts.
  - Do attempt the questions beforehand!
- **Break:** We'll take the attendance before the break.
- **Problem Set:** Hints for current PS, or debrief for previous PS.
- **Lab:** Solve one medium (or two easy) problems.
  - Not graded, but very beneficial to understand CS2040S materials.
  - Hints will be provided to to get (near) Accepted solutions.
  - If you already happen to solve them, you are free to leave the session or stay back to help your peers.

Agenda

For this session:

- Recap first few sessions of CS2040S: Introduction, Basic C++, Basic Analysis of Algorithms.
- Ensuring all students can write simple Java programs using their own devices.
  - Bring the device you intend to use for the practical exams.
- Brief peak in to Abstract Data Types (ADT), which will be taught later this week.

Question 1

ListArray<T>

Question 1a

```
class ListArray<T> { ← (1a)
```

What does this line mean?

Question 1a

```
class ListArray<T> { ← (1a)
```

What does this line mean?

Defines a generic class ListArray.

The file name needs to be `ListArray.java`.

Question 1a: Introduction to Java Generics

- Declare by including a `<TypeParameter>` after class name or before method declaration.
- The `TypeParameter` is only used in compile time, for type checking.
- Compiler will remove all `TypeParameter` and replace by its *erasure*.
  - *Erasure* is the common denominator of the types that the generic code will accept.
- Runtime only knows the *erasure*, not the `TypeParameter`.
- See demo with `GenericDemo.java`.

Question 1b

```
class ListArray<T> {
    private int N;
    private Object[] A = new Object[100]; ← (1b)
```

Anything wrong with this line?

Question 1b

```
class ListArray<T> {
    private int N;
    private Object[] A = new Object[100]; ← (1b)
```

Anything wrong with this line?

No. But we won't be able to add more than 100 elements.

Question 1b

```
class ListArray<T> {
    private int N;
    ~~private Object[] A = new Object[100];~~ ~~← (1b)~~
    private T[] A = new T[100]; ← (extra)
```

Anything wrong with this line?

No. But we won't be able to add more than 100 elements.

Extra Question: Is this better?

Question 1b

```
class ListArray<T> {
    private int N;
    private Object[] A = new Object[100]; ← (1b)
    private T[] A = new T[100];
```

Anything wrong with this line?

No. But we won't be able to add more than 100 elements.

Extra Question: Is this better?

Does not compile! **new** is a runtime operation, and it needs to know the type **T** at runtime.
Remember that runtime doesn't know what **T** is.

Question 1c

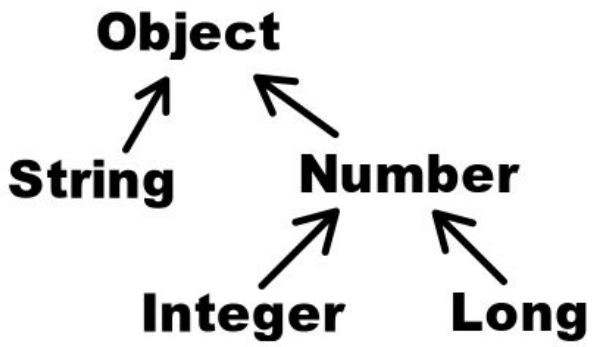
```
class ListArray<T> {
    private int N;
    private Object[] A = new Object[100];

    @SuppressWarnings("unchecked")
    public T get(int i) {
        return (T) A[i]; ← (1c)
    }
```

Anything wrong with this line?

Question 1c: Introduction to Java Type Casting

- A supertype variable can reference a subtype object.
- Compiler doesn't see a problem with casting T to S or vice versa, if either of them is a subtype of another.
  - Compiler will perform this check at compile time.
- However, the cast *may* fail at runtime throwing `ClassCastException`.
  - Because the parent type may be referencing a type from different branch.
- See `CastDemo.java`.



Question 1c

```
class ListArray<T> {
    private int N;
    private Object[] A = new Object[100];

    @SuppressWarnings("unchecked")
    public T get(int i) {
        return (T) A[i]; ← (1c)
    }
```

Anything wrong with this line?

No.

Question 1c

```
class ListArray<T> {
    private int N;
    private Object[] A = new Object[100];

    @SuppressWarnings("unchecked")
    public T get(int i) {
        return (T) A[i]; ← (1c)
    }
```

Anything wrong with this line?

No.

Remember that type casting is both a compile and runtime operation. Here compiler cannot do the compile time checks, because it doesn't know what T the user will put in!

So it raises the unchecked cast warning; can be skipped with `SuppressWarnings`. Then we are to ensure that this cast doesn't fail runtime. Compiler won't perform the pre-checks!

Question 1d

```
public void add(T v) { ← (1d)
    if (N == 100)
        return;
    A[N++] = v;
}
public void add(int i, T v) {
    if (N == 100 || i < 0 || i > N)
        return;
    for (int j = N-1; j >= i; --j)
    // for (int j = i; j <= N-1; ++j)
        A[j+1] = A[j];
    A[i] = v;
    ++N;
}
```

What is the difference of this ‘add’ versus the other ‘add’?

Question 1d

```
public void add(T v) { ← (1d)
    if (N == 100)
        return;
    A[N++] = v;
}
public void add(int i, T v) {
    if (N == 100 || i < 0 || i > N)
        return;
    for (int j = N-1; j >= i; --j)
    // for (int j = i; j <= N-1; ++j)
        A[j+1] = A[j];
    A[i] = v;
    ++N;
}
```

What is the difference of this ‘add’ versus the other ‘add’?

This add adds to the end. Other one adds to position `i`.

Method Overloading: Two or more methods can have same name but different signatures.

Return type is not part of signature.

Question 1e

```
public void add(T v) {
    if (N == 100)
        return;
    A[N++] = v;
}
public void add(int i, T v) {
    if (N == 100 || i < 0 || i > N) ← (1e)
        return;
    for (int j = N-1; j >= i; --j)
    // for (int j = i; j <= N-1; ++j)
        A[j+1] = A[j];
    A[i] = v;
    ++N;
}
```

What does this line mean?

Question 1e

```
public void add(T v) {
    if (N == 100)
        return;
    A[N++] = v;
}
public void add(int i, T v) {
    if (N == 100 || i < 0 || i > N) ← (1e)
        return;
    for (int j = N-1; j >= i; --j)
    // for (int j = i; j <= N-1; ++j)
        A[j+1] = A[j];
    A[i] = v;
    ++N;
}
```

What does this line mean?

Guard clause. If any of these hold, then we cannot actually insert at position `i`.

Question 1f

```
public void add(T v) {
    if (N == 100)
        return;
    A[N++] = v;
}
public void add(int i, T v) {
    if (N == 100 || i < 0 || i > N)
        return;
    for (int j = N-1; j >= i; --j)
    // for (int j = i; j <= N-1; ++j) ← (1f)
        A[j+1] = A[j];
    A[i] = v;
    ++N;
}
```

Can we use this line instead?

Question 1f

```
public void add(T v) {
    if (N == 100)
        return;
    A[N++] = v;
}
public void add(int i, T v) {
    if (N == 100 || i < 0 || i > N)
        return;
    for (int j = N-1; j >= i; --j)
    // for (int j = i; j <= N-1; ++j) ← (1f)
        A[j+1] = A[j];
    A[i] = v;
    ++N;
}
```

Can we use this line instead?

No. It will copy `A[i]` to every `A[i..N]` and destroy our array.

Question 1g

```
public void remove(int i) {
    for (int j = i; j < N-1; ++j) ← (1g)
        A[j] = A[j+1];
    --N;
}
```

Any potential issue with this line?

Question 1g

```
public void remove(int i) {
    for (int j = i; j < N-1; ++j) ← (1g)
        A[j] = A[j+1];
    --N;
}
```

Any potential issue with this line?

No.

Question 1g

```
public void remove(int i) {
    for (int j = i; j < N-1; ++j) ← (1g)
        A[j] = A[j+1];
    --N;
}
```

Any potential issue with this line?

No.

C++ has unsigned integers (Java doesn't). It is common to store sizes as unsigned integers, since it is never negative.

If N was unsigned and 0, then N - 1 would be the maximum representable number, 4294967295 for 32-bits.

Question 1h

```
public void printList() {
    for (int i = 0; i < N; ++i)
        System.out.print(
            (i > 0 ? " " : "") + A[i] ← (1h)
        );
    System.out.println();
}
```

What does this line mean?

Question 1h

```
public void printList() {
    for (int i = 0; i < N; ++i)
        System.out.print(
            (i > 0 ? " " : "") + A[i] ← (1h)
        );
    System.out.println();
}
```

What does this line mean?

?: is called a ternary operator.

`A ? B : C` evaluates to B if A is true, otherwise C.

Same as python's `B if A else C`.

Also note that `A[i]` is implicitly converted to a `String`.

Question 1i

```
public void sortList() {
    // sort array A  ← (1i)
}
```

Implement this method using any sorting algorithm you know!

Question 1i

```
public void sortList() {
    // sort array A  ← (1i)
}
```

Implement this method using any sorting algorithm you know!

Possible solutions:

- Bubble sort
- Insertion sort
- Merge sort
- `Arrays.sort(A, 0, N)`
- etc

Question 2/3

Analysis/ Order of Growth

Complexity analysis

- Is a rough estimate of how execution time will grow with size of input. I.E. *Order of growth.*
- *Time complexity* is commonly used as a metric for comparing the performance of different algorithms on the same task.
- The expression ignores additive or multiplicative constants, not exponents. Eg. $O(100N + 100) = O(N)$, $O(2^N) \mathrel{!=} O(2^{2N})$.
- Just retain the *largest power* for each variable in the expression. Eg. $O(N^3 + N^2) = O(N^3)$.
- When order of growth is applied to measure memory consumption of algorithms, we call that *space complexity.*

Real life application?

Single thread computer: about 10^8 operations/sec

Sorting **N = 10^6** numbers

$O(N^2)$ ~ 2.5 hours

O(N log N) ~ 0.2 sec

Some common big-O terms ordered from slowest growth to fastest growth:

$$O(1) < O(\log N) < O(\sqrt{N}) < O(N) < O(N \log N) < O(N^2) < O(N^3) < O(2^n) < O(N!) < O(N^N)$$

Question 2

Q2). What is the bound of the following function? $\mathbf{F}(n) = \log(\mathbf{2}^n) + \sqrt{n} + \mathbf{100\,000\,000}$

1. $O(n)$
2. $O(n \log n)$
3. $O(n^2)$
4. $O(1)$
5. $O(2^n)$

Question 2

Q2). What is the bound of the following function? $\mathbf{F}(n) = \log(\mathbf{2}^n) + \sqrt{n} + \mathbf{100\,000\,000}$

1. $O(n)$
2. $O(n \log n)$
3. $O(n^2)$
4. $O(1)$
5. $O(2^n)$

$\log(2^n) = n \log(2)$

$\log(2^n) + \text{sqrt}(n) + 100000000$
$= n \log 2 + n^{1/2} + 100000000$

Question 2

Q2). What is the bound of the following function? $\mathbf{F}(n) = \log(\mathbf{2}^n) + \sqrt{n} + \mathbf{100\,000\,000}$

1. $O(n)$
2. $O(n \log n)$
3. $O(n^2)$
4. $O(1)$
5. $O(2^n)$

$\log(2^n) = n \log(2)$

$\log(2^n) + \text{sqrt}(n) + 100000000$

$= n \log 2 + n^{1/2} + 100000000$

$= n \cancel{\log 2} + \cancel{n^{1/2}} + \cancel{100000000}$

$= O(n)$

Question 3.a

Q3.a). What is the bound of the following function? $\mathbf{F}(n) = n + \frac{1}{2}n + \frac{1}{3}n + \frac{1}{4}n + \ldots + \mathbf{1}$

1. $O(2^n)$
2. $O(n^2)$
3. $O(n \log n)$
4. $O(n)$
5. $O(\log^2 n)$
6. $O(\log n)$
7. $O(1)$
8. none of the above

Question 3.a

$$F(n) = n + \tfrac{1}{2} n + \tfrac{1}{3} n + \tfrac{1}{4} n + \ldots + 1$$

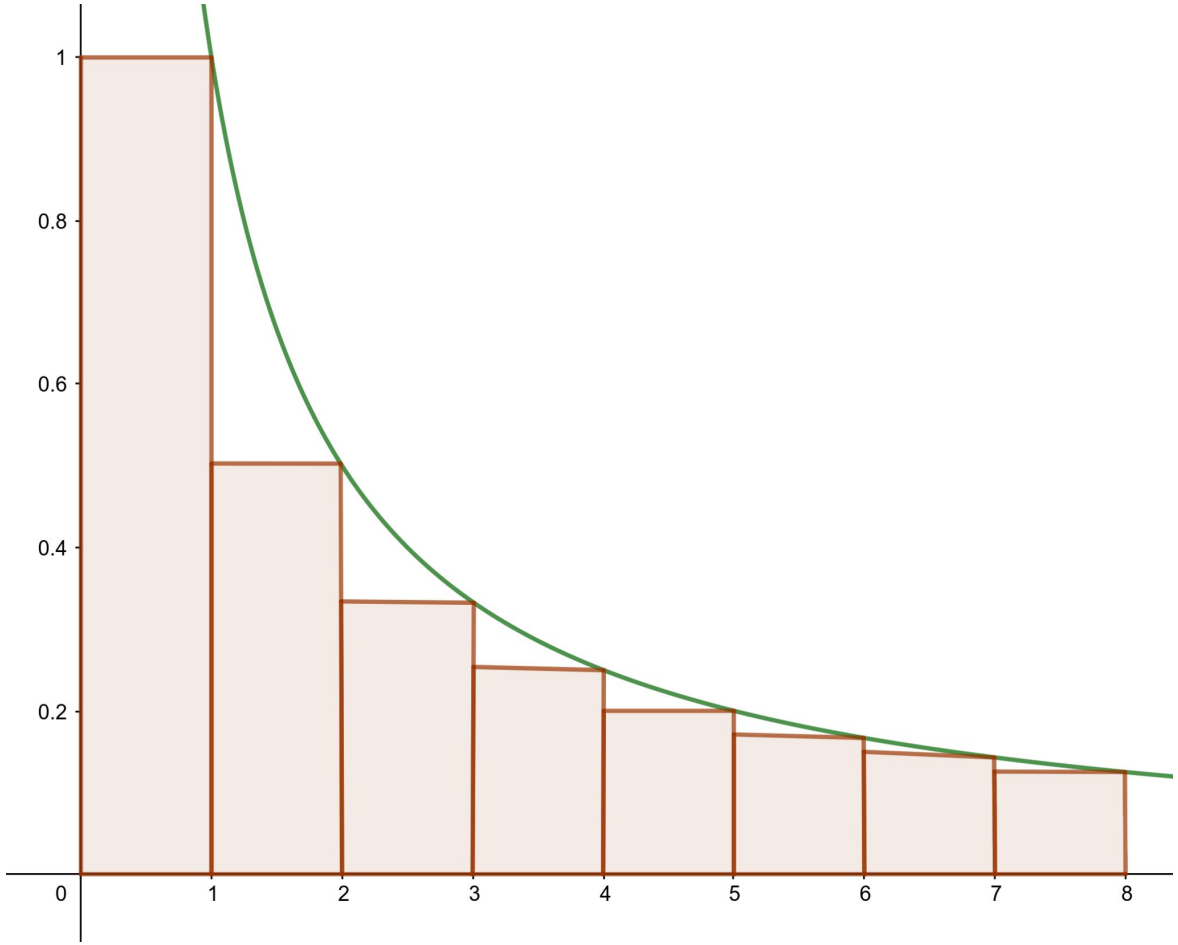
$$O(F(n)) = O(n + \tfrac{1}{2} n + \tfrac{1}{3} n + \tfrac{1}{4} n + \ldots + 1)$$

$$= O(n\,(\underbrace{1 + \tfrac{1}{2} + \tfrac{1}{3} + \tfrac{1}{4} + \ldots + 1/n}_{\text{How to deal with this?}}))$$

Question 3.a: Harmonic Series

Using integration:

$$
\begin{aligned}
& 1 + \frac{1}{2} + \cdots + \frac{1}{n} \\
&= 1 + \sum_{i=2}^{n} \frac{1}{i} \\
&< 1 + \int_{1}^{n} \frac{1}{x}\, dx \\
&= 1 + \ln(n)
\end{aligned}
$$



Question 3.a: Solution

From Harmonic Series

$$O(1 + \tfrac{1}{2} + \tfrac{1}{3} + \tfrac{1}{4} + \ldots + 1/n) = O(\log n)$$

$$O(F(n)) = O(n\,(1 + \tfrac{1}{2} + \tfrac{1}{3} + \tfrac{1}{4} + \ldots + 1/n))$$

$$= \mathbf{O(n \log n)}$$

Question 3.b

Q3.b). What about $\mathbf{G}(n) = n + \frac{1}{2}n + \frac{1}{4}n + \frac{1}{8}n + \ldots + 1$

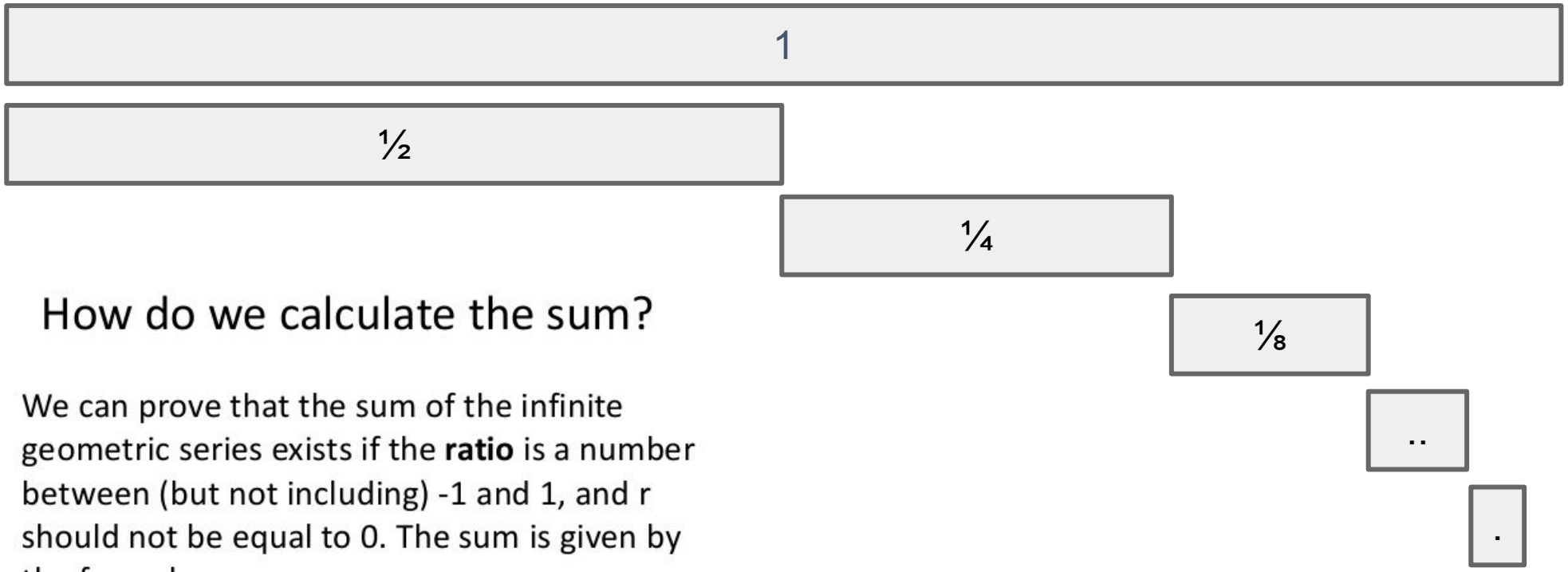
Question 3.b

$$G(n) = n + \tfrac{1}{2} n + \tfrac{1}{4} n + \tfrac{1}{8} n + \ldots + 1$$

$$O(G(n)) = O(n + \tfrac{1}{2} n + \tfrac{1}{4} n + \tfrac{1}{8} n + \ldots + 1)$$

$$= O(n\,(\underbrace{1 + \tfrac{1}{2} + \tfrac{1}{4} + \tfrac{1}{8} + \ldots + 1/n}_{\text{How to deal with this?}}))$$

Question 3.b: Convergent Geometric Series



How do we calculate the sum?

We can prove that the sum of the infinite geometric series exists if the **ratio** is a number between (but not including) -1 and 1, and r should not be equal to 0. The sum is given by the formula:

$$\sum_{k=0}^{\infty} ar^k = a + ar + ar^2 + ar^3 + \cdots = \frac{a}{1-r}$$

Question 3.b: Solution

From Convergent Geometric Series

$$O(1 + \tfrac{1}{2} + \tfrac{1}{4} + \tfrac{1}{8} + \ldots + 1/n) = O(1 / (1 - \tfrac{1}{2}))$$

$$= O(2) = O(1)$$

$$O(G(n)) = O(n\,(1 + \tfrac{1}{2} + \tfrac{1}{4} + \tfrac{1}{8} + \ldots + 1/n))$$

$$= \mathbf{O(n)}$$

Question 3.c

Q3.c). $\mathbf{H}(n) = n + \frac{1}{2}n + \frac{1}{3}n + \frac{1}{5}n + \frac{1}{7}n + \frac{1}{11}n + \frac{1}{13}n + \frac{1}{17}n + \ldots + \mathbf{1}$

$\mathbf{H}(n)$ is basically the sum of the reciprocals of prime numbers up to n.

Question 3.c: Solution

```
F: 1/1 1/2 1/3 1/4 1/5 1/6 1/7 1/8 1/9 1/10 1/11 1/12 ...
H: 1/1 1/2 1/3     1/5     1/7             1/11
G: 1/1 1/2     1/4             1/8 ... ... ... ... ... 1/16
```

Question 3.c: Solution

```
F: 1/1 1/2 1/3 1/4 1/5 1/6 1/7 1/8 1/9 1/10 1/11 1/12 ...
H: 1/1 1/2 1/3     1/5     1/7             1/11
G: 1/1 1/2     1/4             1/8 ... ... ... ... ... 1/16
```

- H only has a subset of terms of **F** $\Rightarrow$ **H < F**.

Question 3.c: Solution

```
F: 1/1 1/2 1/3 1/4 1/5 1/6 1/7 1/8 1/9 1/10 1/11 1/12 ...
H: 1/1 1/2 1/3     1/5     1/7                 1/11
G: 1/1 1/2     1/4                 1/8 ... ... ... ... ... 1/16
```

- H only has a subset of terms of **F** ⇒ **H < F**.
- H has at least one term between **1/n** and **1/2n** bigger than **1/2n** ⇒ **G < H**.
  - **Bertrand's postulate:** There is always a prime between **n** and **2n** for any **n**.

Question 3.c: Solution

```
F: 1/1 1/2 1/3 1/4 1/5 1/6 1/7 1/8 1/9 1/10 1/11 1/12 ...
H: 1/1 1/2 1/3     1/5     1/7                 1/11
G: 1/1 1/2     1/4                 1/8 ... ... ... ... ... 1/16
```

- H only has a subset of terms of **F** ⇒ **H < F**.
- H has at least one term between **1/n** and **1/2n** bigger than **1/2n** ⇒ **G < H**.
  - **Bertrand's postulate:** There is always a prime between **n** and **2n** for any **n**.
- In the given options, there is nothing between **F = O(n)** and **G = O(n lg n)**.

Question 3.c: Solution

```
F: 1/1 1/2 1/3 1/4 1/5 1/6 1/7 1/8 1/9 1/10 1/11 1/12 ...
H: 1/1 1/2 1/3     1/5     1/7                 1/11
G: 1/1 1/2     1/4                 1/8 ... ... ... ... ... 1/16
```

- H only has a subset of terms of **F** ⇒ **H < F**.
- H has at least one term between **1/n** and **1/2n** bigger than **1/2n** ⇒ **G < H**.
  - **Bertrand's postulate:** There is always a prime between **n** and **2n** for any **n**.
- In the given options, there is nothing between **F = O(n)** and **G = O(n lg n)**.
- **(Out of Syllabus)** Sum of reciprocals of primes up to **n** is **O(log log n)**. So the actual answer is **O(n log log n)**, which is indeed between **O(n)** and **O(n lg n)**.

If have time

Additional questions

Example (AY17/18 S1 Midterm Paper)

If have time

What's the time complexity?

```
int N, counter = 0;
cin >> N;
for (int i = N; i >= 1; i--) {
    for (int j = 1; j <= N/i; j++) {
        counter++;
    }
}
cout << counter << endl;
```

If have time

What's the time complexity?

```
int N, counter = 0;
cin >> N;
for (int i = N; i >= 1; i--) {
    for (int j = 1; j <= N/i; j++) {
        counter++;
    }
}
cout << counter << endl;
```

O(N log N)

If have time

What's the time complexity?

```
int N, counter = 0;
cin >> N;
for (int i = 0; i < N; i++) {
    for (int j = 0; j < i; j++) {
        counter += j;
    }
}
cout << counter << endl;
```

If have time

What's the time complexity?

```
int N, counter = 0;
cin >> N;
for (int i = 0; i < N; i++) {
    for (int j = 0; j < i; j++) {
        counter += j;
    }
}
cout << counter << endl;
```

$O(N^2)$

If have time

What's the time complexity?

```
int N, counter = 0;
cin >> N;
for (int i = 0; i < N; i++) {
    for (int j = 0; j < N; j++) {
        counter++;
        i++;
    }
}
cout << counter << endl;
```

If have time

What's the time complexity?

```
int N, counter = 0;
cin >> N;
for (int i = 0; i < N; i++) {
    for (int j = 0; j < N; j++) {
        counter++;
        i++;
    }
}
cout << counter << endl;
```

$O(N)$

Attendance
Break
Questions

PS1 Debrief

/tenis Solution

- Mostly a statement parsing problem. Need to make lots of inferences.

/tenis Solution

- Mostly a statement parsing problem. Need to make lots of inferences.
- **A player wins a set if he wins 6 or more games and at least two games more than his opponent.**

/tenis Solution

- Mostly a statement parsing problem. Need to make lots of inferences.
- **A player wins a set if he wins 6 or more games and at least two games more than his opponent.**
  - A set cannot be won with less than 6 games. 1:3, 2:4, 3:5 etc are invalid.

/tenis Solution

- Mostly a statement parsing problem. Need to make lots of inferences.
- **A player wins a set if he wins 6 or more games and at least two games more than his opponent.**
  - A set cannot be won with less than 6 games. 1:3, 2:4, 3:5 etc are invalid.
  - A set cannot have both player getting same number of wins. 1:1, 2:2 etc are invalid.

/tenis Solution

- Mostly a statement parsing problem. Need to make lots of inferences.
- **Additionally, if the result is 6:6 in the first or second set (but not the third set), a single final game is played to determine the winner of the set (the tie-break game).**

/tenis Solution

- Mostly a statement parsing problem. Need to make lots of inferences.
- **Additionally, if the result is 6:6 in the first or second set (but not the third set), a single final game is played to determine the winner of the set (the tie-break game).**
  - In first two set: whenever 6:6 is reached, only one more game is played and the set ends in 6:7 or 7:6. Only place a player can win by 1 game.

/tenis Solution

- Mostly a statement parsing problem. Need to make lots of inferences.
- **Additionally, if the result is 6:6 in the first or second set (but not the third set), a single final game is played to determine the winner of the set (the tie-break game).**
  - In first two set: whenever 6:6 is reached, only one more game is played and the set ends in 6:7 or 7:6. Only place a player can win by 1 game.
  - Infer that first two games cannot have a number > 7.

/tenis Solution

- Mostly a statement parsing problem. Need to make lots of inferences.
- **Additionally, if the result is 6:6 in the first or second set (but not the third set), a single final game is played to determine the winner of the set (the tie-break game).**
  - In first two set: whenever 6:6 is reached, only one more game is played and the set ends in 6:7 or 7:6. Only place a player can win by 1 game.
  - Infer that first two games cannot have a number > 7.
  - The third game however, can go forever, until a player has 2 more wins.

PS1: /tenis Solution

- Mostly a statement parsing problem. Need to make lots of inferences.
- **The match ends when either player has won 2 sets. That player is the winner.**

PS1: /tenis Solution

- Mostly a statement parsing problem. Need to make lots of inferences.
- **The match ends when either player has won 2 sets. That player is the winner.**
  - By previous inferences, there are no ties in a set.

PS1: /tenis Solution

- Mostly a statement parsing problem. Need to make lots of inferences.
- **The match ends when either player has won 2 sets. That player is the winner.**
  - By previous inferences, there are no ties in a set.
  - Therefore, by this point, there cannot be more than 3 sets!

PS1: /falcondrive Solution

- Scan both images in row major order. The first silhouette pixels we hit in both images are correspondances.
- Take the two pixels' difference to calculate the deviations in X/Y axis.
- Merge both images to form the background.
- Paint the new silhouette over the background.

PS2 Discussion

PS2: /universityzoning

- Reading comprehension! Let’s try to unpack.

PS2: /universityzoning

- Reading comprehension! Let's try to unpack.
- **R** rows, **C** columns, **F** faculties, **S** students, **G** threshold of compliant faculty.

PS2: /universityzoning

- Reading comprehension! Let's try to unpack.
- **R** rows, **C** columns, **F** faculties, **S** students, **G** threshold of compliant faculty.
- The students of a faculty are assigned cells in the faculty in row major order, in increasing order of their IDs.

PS2: /universityzoning

- Reading comprehension! Let's try to unpack.
- **R** rows, **C** columns, **F** faculties, **S** students, **G** threshold of compliant faculty.
- The students of a faculty are assigned cells in the faculty in row major order, in increasing order of their IDs.
- A faculty **f** is *compliant* if at least $\mathbf{T_f}$ students are at their assigned spot.

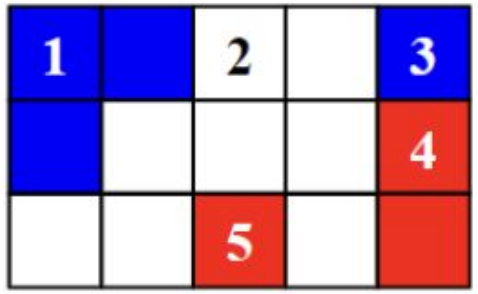
PS2: /universityzoning

- Reading comprehension! Let's try to unpack.
- **R** rows, **C** columns, **F** faculties, **S** students, **G** threshold of compliant faculty.
- The students of a faculty are assigned cells in the faculty in row major order, in increasing order of their IDs.
- A faculty **f** is *compliant* if at least $\mathbf{T_f}$ students are at their assigned spot.
- Task: Calculate minimum number of steps the students needs to take, to have at least **G** compliant faculties.

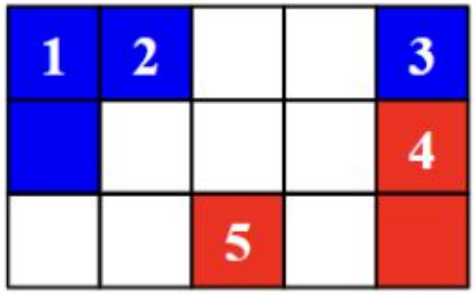
PS2: /universityzoning

Faculty 1 **(Blue)**
Student ID: 1, 2, 3

Faculty 2 **(Red)**
Student ID: 4, 5



Initial State



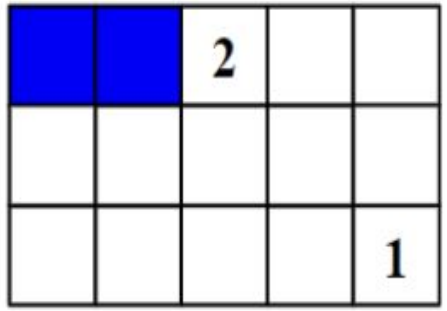
Solution

Sample Input 1

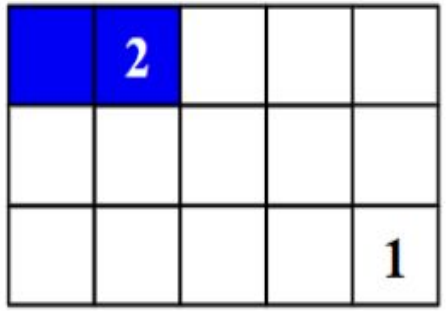
```
3 5 2 5 2
4 1 1 1 2 1 5 2 1
3 2 5 3 3 3 5
1 1 1 1
1 3 2 1
1 5 3 1
2 5 4 2
3 3 5 2
3 2
```

PS2: /universityzoning

Faculty 1 **(Blue)**
Student ID: 1, 2



Initial State



Solution

Sample Input 2

```
3 5 1 2 1
2 1 1 1 2
3 5 1 1
1 3 2 1
1
```

PS2: /jobbyte

- Minimum number of swaps needed to sort an array.
- Observation: Optimal as long as each swap fixes at least one index.
- This gives an $O(N^2)$ solution: Perform insertion sort and count # swaps.
- Figure out another efficient scheme of swapping!

Hands-on session

Lab: /cutinline

- Hints will be posted at 5 minutes intervals.

Lab: /cutinline

- Hints will be posted at 5 minutes intervals.
- 5 mins: Read ArrayList<E> documentation.

Lab: /cutinline

- Hints will be posted at 5 minutes intervals.
- 5 mins: Read ArrayList<E> documentation.
- 10 mins: Use `indexOf(Object)` and `remove()` for `leave`.

Lab: /cutinline

- Hints will be posted at 5 minutes intervals.
- 5 mins: Read ArrayList<E> documentation.
- 10 mins: Use `indexOf(Object)` and `remove()` for `leave`.
- 15 mins: Use `add(int index, E element)`.
- That should be all needed to solve this problem!

Thank You!

Anonymous Feedback:
https://forms.gle/MkETeXdUT53Vhh896