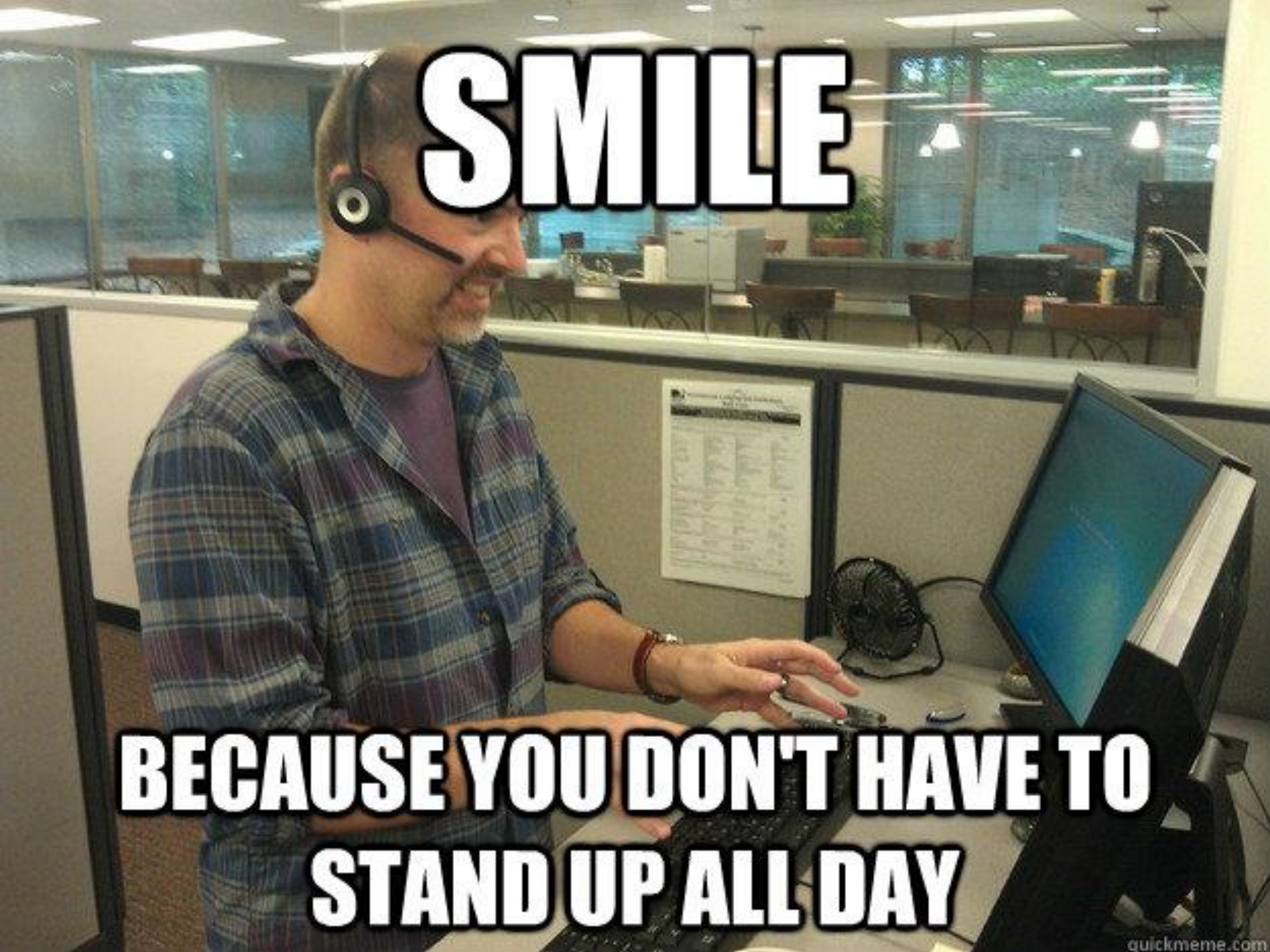
Tutorial –2
Last tutorial before PE2


SMILE
BECAUSE YOU DON'T HAVE TO STAND UP ALL DAY
quickmeme.com

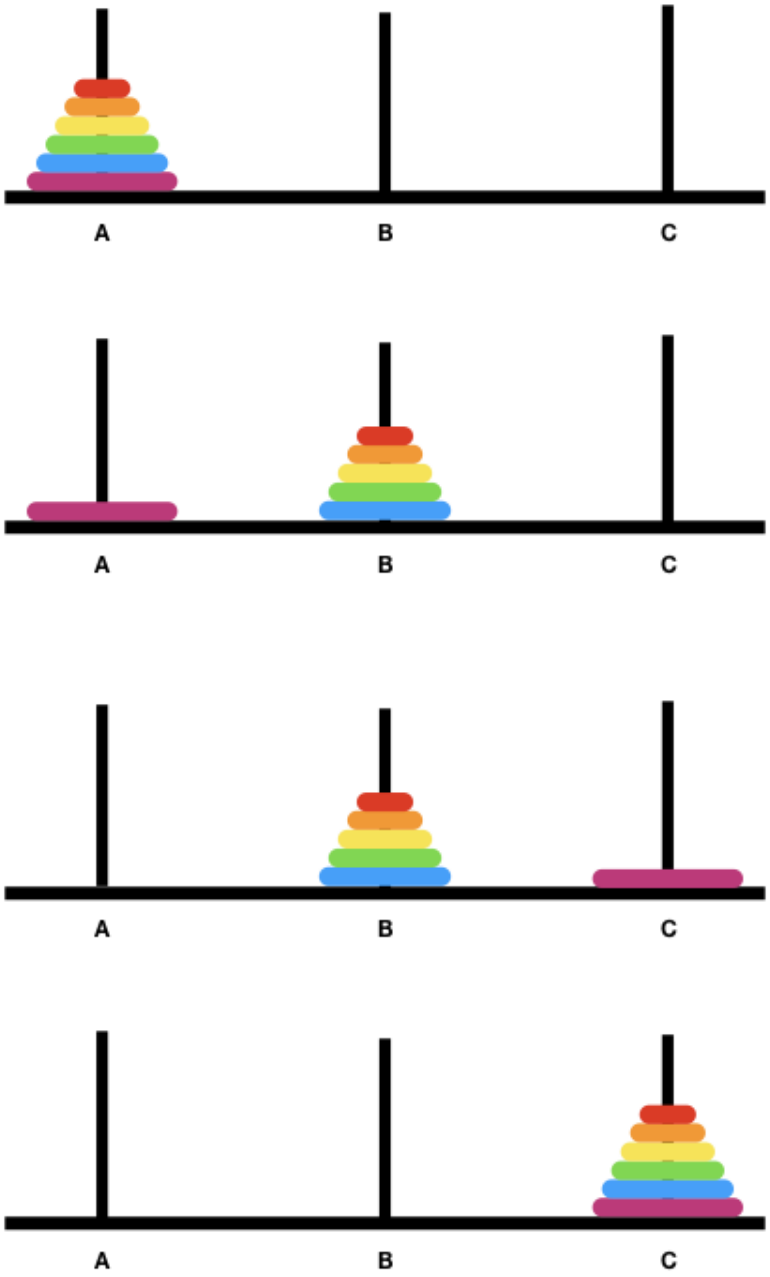
Tutorial 10

Xiao Liang's Tutorial Group

5 November 2018

https://github.com/yxliang01/cs1010-fun-stuff

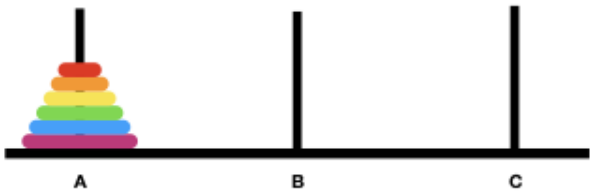
Problem 25.1


A
B
C
A
B
C
A
B
C
A
B
C

```
void solve(long k, long source, long dest, long
placeholder) {
  if (k == 1) {
    print(k, source, dest);
  }
  else {
    solve(k - 1, source, placeholder, dest);
    print(k, source, dest);
    solve(k - 1, placeholder, dest, source);
  }
}
```

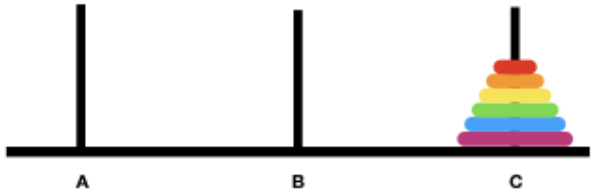
Only allowed to move a disk either to Peg B from another peg or from Peg B to another peg.

Solution


A
B
C


A
B
C


A
B
C


A
B
C

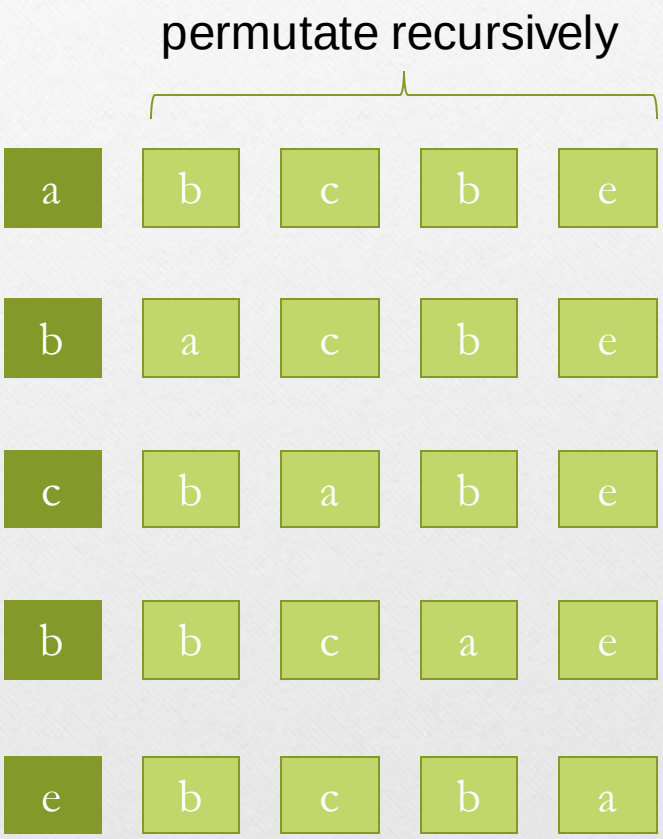

A
B
C


A
B
C

```
void solve(long k, long source, long dest, long placeholder) {
  if (k == 1) {
    move(k, source, placeholder);
    move(k, placeholder, dest);
    return;
  }
  solve(k - 1, source, dest, placeholder);
  move(k, source, placeholder);
  solve(k - 1, dest, source, placeholder);
  move(k, placeholder, dest);
  solve(k - 1, source, dest, placeholder);
}
```

Problem 26.1

Code given in lecture



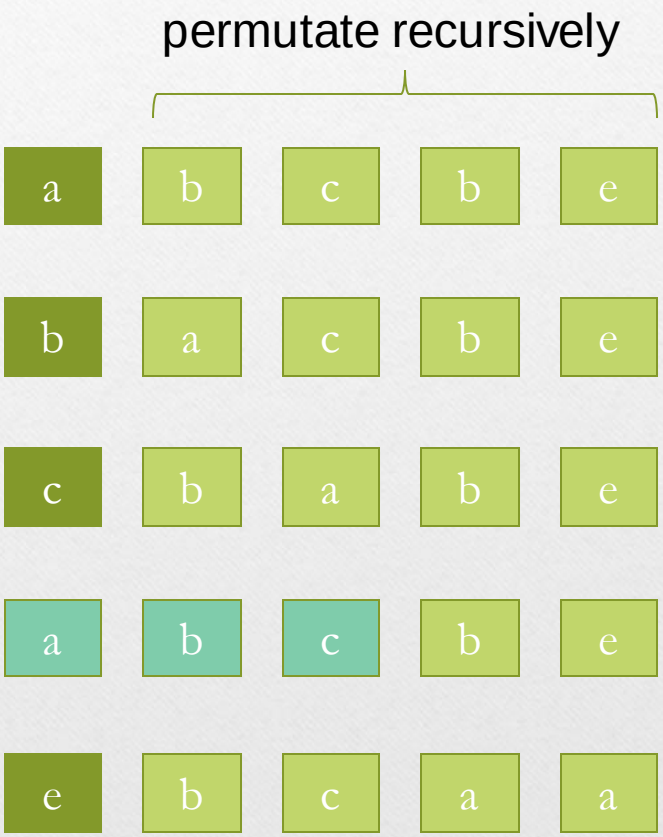
duplicate work!

Avoid generating duplicate permutations when the input string contains duplicate characters

```
void permute(char a[], long len, long curr) {
 if (curr == len-1) {
   cs1010_println_string(a);
   return;
  }

  permute(a, len, curr + 1);
  for (long i = curr + 1; i < len; i += 1) {
   if (...) { // Line A
     swap(a, curr, i);
     permute(a, len, curr + 1);
     swap(a, i, curr);
   }
  }
}
```

Solution

Code given in lecture



check before recurse:
has it appeared before?
if so, skip


```
bool appear_before(char a[], long k,
                   long i) {
  for (int j = k; j < i; j += 1) {
    if (a[j] == a[i]) {
      return true;
    }
  }
  return false;
}
```

```
void permute(char a[], long len, long curr) {
 if (curr == len-1) {
   cs1010_println_string(a);
   return;
 }

 permute(a, len, curr + 1);
 for (long i = curr + 1; i < len; i += 1) {
   if (!appear_before(a, curr, i)) { // Line A
     swap(a, curr, i);
     permute(a, len, curr + 1);
     swap(a, i, curr);
   }
 }
```

Fun Stuff

Uh?

When ur lurking and see something funny but can't retweet it cause u lurking

Problem 27.1

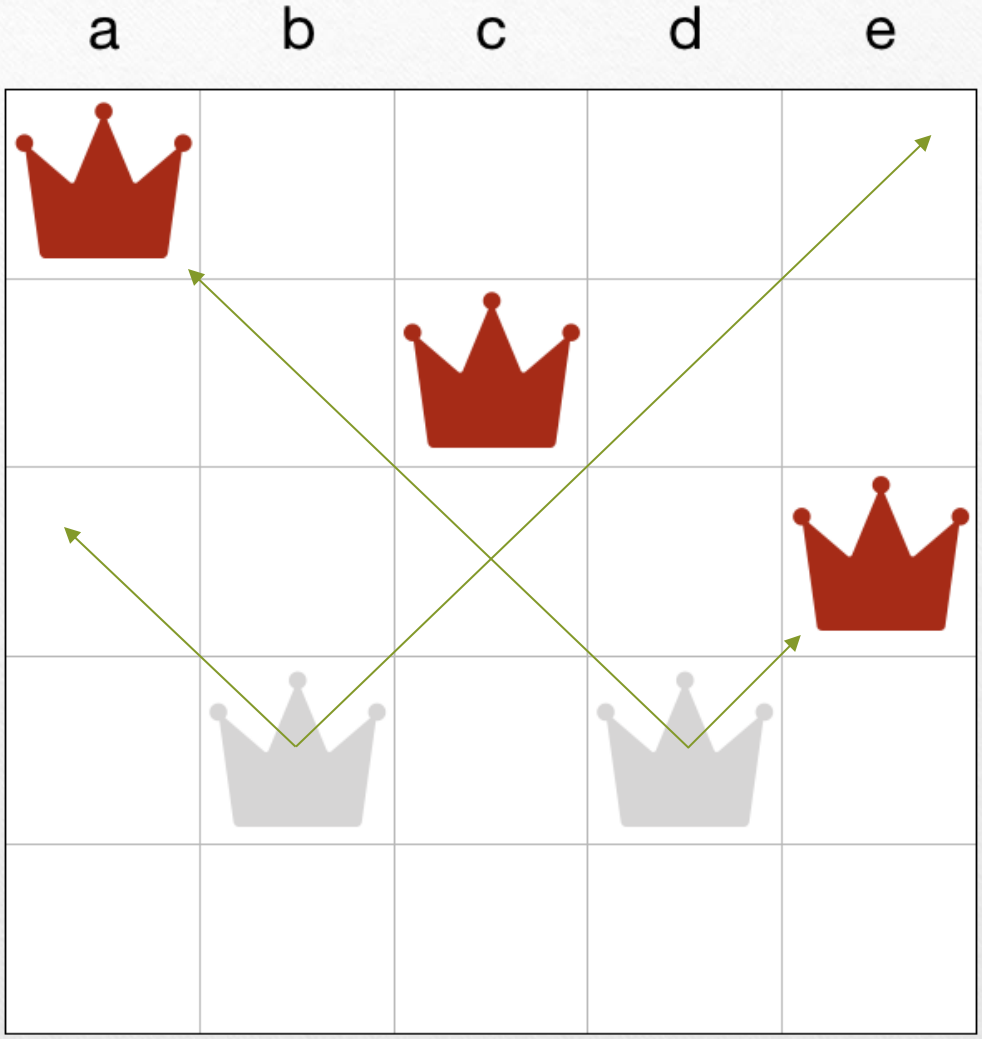
```
void nqueens(char queens[], long n, long row) {
 if (row == n-1) {
  if (!threaten_each_other_diagonally(queens, n)) {
   cs1010_println_string(queens);
  }
  return;
 }

 if (!threaten_each_other_diagonally(queens, row)) {
  nqueens(queens, n, row + 1);
 }
 for (long i = row + 1; i < n; i++) {
  swap(queens, row, i);
  if (!threaten_each_other_diagonally(queens, row)) {
   nqueens(queens, n, row + 1);
  }
  swap(queens, row, i);
 }
}
```

```
bool threaten_each_other_diagonally(char queens[], long len) {
  for (long i = 0; i < len; i += 1) {
    // for each queen in row i, check rows i+1 onwards,
    // on both left (-=1) and right (+=1) side, if there
    // is a queen in that column.
    if (has_a_queen_in_diagonal(queens, len, i)) {
      return true;
    }
  }
  return false;
}
```

Identify the repetitive work being done in the calls threaten_each_other_diagonally.

Suggest a way to remove the repetitive work.

Solution



Already "safe." No need to check again.

Only check the new row against the queens above.

Problem 27.2

```
void permute(char a[], long len, long curr) {
 if (curr == len-1) {
   cs1010_println_string(a);
   return;
 }

 permute(a, len, curr + 1);
 for (long i = curr + 1; i < len; i += 1) {
   if (!appear_before(a, curr, i)) { // Line A
     swap(a, curr, i);
     permute(a, len, curr + 1);
     swap(a, i, curr);
   }
 }
}
```

Restrict the permutations to those where the same character does not appear next to each other.

```
void permute(char a[], long len, long curr) {
 if (curr == len-1) {
   cs1010_println_string(a);
   return;
 }

 permute(a, len, curr + 1);
 for (long i = curr + 1; i < len; i += 1) {
   if (!appear_before(a, curr, i)) { // Line A
     swap(a, curr, i);
     permute(a, len, curr + 1);
     swap(a, i, curr);
   }
 }
```

Solution

Prune away generation of permutations where the same characters appear next to each other.

```
void permute(char a[], long len, long curr) {
 if (curr == len-1) {
   if (a[curr] != a[curr-1]) {
     cs1010_println_string(a);
   }
   return;
 }
 if (a[curr] != a[curr-1]) {
   permute(a, len, curr + 1);
 }
 for (long i = curr + 1; i < len; i += 1) {
   if (!appear_before(a, curr, i) && a[i] != a[curr-1]) {
     swap(a, curr, i);
     permute(a, len, curr + 1);
     swap(a, i, curr);
   }
 }
}
```

- Good Luck (English)
- 祝你好运 (华语，Mandarin)
- 祝你好运 (闽南语, Hokkien)
- 祝你好運 (廣東話, Cantonese)
- Semoga Berjaya (Bahasa Melayu, Malay)
- Semoga Berhasil(Bahasa Indonesia)
- நல்ல அதிர்ஷ்டம் (தமிழ், Tamil)
- ? (हिन्दी, Hindi)
- がんばろう（日本語, Japanese）

This slide is at

https://github.com/yxliang01/cs1010-fun-stuff/blob/master/tutorial-slides/tut10-xl.pptx

PDF version

https://github.com/yxliang01/cs1010-fun-stuff/blob/master/tutorial-slides/tut10-xl.pdf

Q&A