Завдання №1

# ДЕШИФРУВАТИ ПОВІДОМЛЕННЯ, ЗАШИФРОВАНЕ ШИФРОМ БЕКОНА

Для кодування повідомлень Френсіс Бекон запропонував кожну літеру тексту замінювати на групу з п'яти символів «A» або «B» (назвемо їх "ab-групами"). Для співставлення літер і кодуючих ab-груп в даному завданні використовується ключ-
ланцюжок **aaaaabbbbbabbbaabbababbaaababaab**, в якому порядковий номер літери відповідає порядковому номеру початку ab-групи.

Наприклад: літера "a" - перша літера алфавіту; для визначення її коду беремо 5 символів з ключа, починаючи з першого: aaaaa. Літера "c" - третя в алфавіті, отже для визначення її коду беремо 5 символів з ключа, починаючи з третього: aaabb.

Таким чином, оригінальне повідомлення перетворюється на послідовність ab-груп і може далі бути накладене на будь-який текст відповідної довжини: A позначається нижнім регістром, B - верхнім.

Наприклад, початкове повідомлення - Prometheus.

1. Кодуємо його через ab-послідовності:
p = abbab
r = babab
o = aabba
m = bbaab
e = abbbb
t = babba
h = bbbab
e = abbbb
u = abbaa
s = ababb
Результат: abbab babab aabba bbaab abbbb babba bbbab abbbb abbaa ababb

2. Підбираємо текст приблизно такої ж довжини, в якому сховаємо наше повідомлення: Welcome to the Hotel California Such a lovely place Such a lovely place

3. Для зручності розбиваємо його на групи по 5 символів і відкидаємо зайву частину (щоб при декодуванні не отримувалися зайві п'ятірки).
Співставимо ab-рядок і текст-сховище для порівняння:
abbab babab aabba bbaab abbbb babba bbbab abbbb abbaa ababb
Welco metot heHot elCal iforn iaSuc halov elypl aceSu chalo vely

4. Змінюємо регістр символів, кодуючи A та B:
abbab babab aabba bbaab abbbb babba bbbab abbbb abbaa ababb
wELcO MeToT heHOt ELcaL iFORN IaSUc HALoV eLYPL aCEsu cHaLO vely

5. Повертаємо початкові пробіли на місце:
wELcOMe To The HOtEL caLiFORNIa SUcH A LoVeLY PLaCE sucH a LOvely

Для дешифрування повідомлення необхідно виконати зворотні дії.

---

**Вхідні дані**: рядок, передається в програму як аргумент командного рядка. Може містити пробіли та літери латинського алфавіту в будь-якому регістрі. Для передачі в якості одного аргументу рядок береться в подвійні лапки.

**Результат роботи**: рядок - дешифроване повідомлення.

**Наприклад**

1. Вхідні дані: I canT DAnCE i CANt TAlK Hey
2. видаляємо пробіли, розбиваємо на групи по 5 символів: IcanT DAnCE iCANt TAlKH ey
3. ey відкдається
4. символи нижнього регістру перетворюються в a, верхнього - в b:
   baaab bbabb abbba bbabb
5. декодуємо, використовуючи ключ:
   baaab = w
   bbabb = i
   abbba = k
   bbabb = i
6. Результат: wiki

1. Вхідні дані: Hot sUn BEATIng dOWN bURNINg mY FEet JuSt WalKIng arOUnD HOt suN mAkiNG me SWeat

2. видаляємо пробіли, розбиваємо на групи по 5 символів: HotsU nBEAT IngdO WNbUR NINgm YFEet JuStW alKIn garOU nDHOt suNmA kiNGm eSWea t
3. t відкдається
4. символи нижнього регістру перетворюються в a, верхнього - в b: baaab abbbb baaab bbabb bbbaa bbbaa babab aabba aaabb abbba aabab aabba abbaa
5. декодуємо, використовуючи ключ:
   baaab = w
   abbbb = e
   baaab = w
   bbabb = i
   bbbaa = l
   bbbaa = l
   babab = r
   aabba = o
   aaabb = c
   abbba = k
   aabab = y
   aabba = o
   abbaa = u
6. Результат: wewillrockyou

Завдання №2

## ІНДІАНА ДЖОНС ТА ПЕРСТЕНЬ ЦЗУНЬ-СИ

На одному східному ринку до рук Індіани Джонса потрапив цікавий документ. У ньому згадувався загадковий перстень, який належав славетному військовому стратегу стародавності Цзунь-Си і, за словами автора, деяким чином увібрав частку його мудрості. Ймовірно, далі перстень передавався від одного імператора Китаю до іншого і врешті-решт був похований разом з одним із них.

Для того, щоб перевірити цю інформацію, необхідно навідатися в імператорські усипальні, які являють собою лабіринт з кімнат і переходів. Влада Китаю не дозволяє проводити там жодних досліджень. Але через знайомого в міністерстві культури доктор Джонс отримав дозвіл на запуск робота-дослідника для пошуку артефакта.

Вам необхідно скласти **функцію maze_controller()** для керування роботом. Відомо, що лабіринт є квадратним, десь в ньому має знаходитися перстень Цзунь-Си і все. План лабіринту та його точні розміри, як і точне місцезнаходження входу та шуканого артефакту, невідомі.

На жаль, замість новітнього ВОЛЛІ-3000 доктору Джонсу продали більш дешевий БАЛЛІ-3000, недоліками якого є дуже обмежений радіус дії сенсорів і відсутність вбудованої функції складання карт. Тому наявний робот "бачить" лише те, що знаходиться лише безпосередньо перед ним і визначає наявність перешкод на дорозі лише при безпосередньому контакті з ними.

У робота є об'єктно-орієнтований інтерфейс управління із наступними методами:

- go() -- проїхати на поле вперед, повертає True або False в залежності від того, чи вдалося проїхати (наприклад, перед роботом може знаходитися стіна). Якщо проїхати неможливо, робот залишається на місці.
- turn_left() -- повернути на 90 градусів проти годинникової стрілки.
- turn_right() -- повернути на 90 градусів за годинниковою стрілкою.
- found() -- перевіряє, чи знаходиться перстень у зоні видимості робота.

В якості єдиного аргументу при виклику функції maze_controller() передається ініціалізований об'єкт класу MazeRunner для управління роботом. Функція maze_controller нічого не повертає за допомогою оператора return. Але в результаті її роботи робот має бути переведений у поле лабіринту, в якому знаходиться шуканий артефакт (вважати, що артефакт завжди наявний в лабіринті). Тобто після виклику maze_controller(maze_runner), метод об'єкту maze_runner.found() повинен повертати True. Прямий доступ до зображення лабіринту заборонено.

Приклад класу MazeRunner можна взяти тут.

Приклад послідовності дій для тестування (наступні дії виконуватимуться автоматичним перевіряльником і не повинні міститися в вашому розв'язку):

**приклад лабіринту №1 (зовсім простий):**

```
maze_example1 = {
    'm': [
        [0,1,0,0,0],
        [0,1,1,1,1],
        [0,0,0,0,0],
        [1,1,1,1,0],
        [0,0,0,1,0],
    ],
    's': (0,0),
    'f': (4,4)
}
maze_runner = MazeRunner(maze_example1['m'], maze_example1['s'],
maze_example1['f']) # ініціалізація робота
maze_controller(maze_runner) # виклик вашої функції
print maze_runner.found()   # перевірка того, що артефакт знайдено, повинно бути
True
```

## приклад лабіринту №2 (простий):

```
maze_example2 = {
    'm': [
        [0,0,0,0,0,0,0,1],
        [0,1,1,1,1,1,1,1],
        [0,0,0,0,0,0,0,0],
        [1,1,1,1,0,1,0,1],
        [0,0,0,0,0,1,0,1],
        [0,1,0,1,1,1,1,1],
        [1,1,0,0,0,0,0,0],
        [0,0,0,1,1,1,1,0],
    ],
    's': (7,7),
    'f': (0,0)
}
maze_runner = MazeRunner(maze_example2['m'], maze_example2['s'],
maze_example2['f'])
maze_controller(maze_runner)
print maze_runner.found()   # True
```

## приклад лабіринту №3 (середньої складності):

```
maze_example3 = {
    'm': [
        [0,0,0,0,0,0,0,0,0,0,0],
        [1,0,1,1,1,0,1,1,1,0,1],
        [1,0,1,0,0,0,0,0,1,0,1],
        [1,0,1,0,1,0,1,0,1,0,1],
        [1,0,1,0,1,0,1,0,1,0,1],
        [1,0,1,0,1,0,1,0,1,0,1],
        [1,0,1,0,1,0,1,0,1,0,1],
        [1,0,1,0,1,0,1,0,1,0,1],
        [1,0,1,0,1,0,1,0,1,0,1],
        [1,0,1,0,1,1,1,0,1,0,1],
        [1,0,1,0,0,0,0,0,1,0,1],
    ],
    's': (0,5),
    'f': (10,5)
}
maze_runner = MazeRunner(maze_example3['m'], maze_example3['s'],
maze_example3['f'])
maze_controller(maze_runner)
print maze_runner.found()   # True
```

## приклад лабіринту №4 (складніший):

```
maze_example4 = {
    'm': [
        [0,0,0,0,1,0,1,0,0,0,0],
        [0,1,1,1,1,0,1,1,1,1,0],
        [0,0,0,0,0,0,0,0,0,0,0],
        [0,1,0,1,1,1,1,1,0,1,0],
        [1,1,0,1,0,0,0,1,0,1,1],
        [0,1,0,1,0,1,0,1,0,1,0],
        [0,1,0,0,0,1,0,0,0,1,0],
        [0,1,0,1,1,1,1,1,0,1,0],
        [0,1,0,0,0,0,0,0,0,1,0],
        [0,1,1,1,1,0,1,1,1,1,0],
        [0,0,0,0,0,0,0,0,0,0,0],
    ],
    's': (0,5),
    'f': (4,5)
```

```
}
maze_runner = MazeRunner(maze_example4['m'], maze_example4['s'],
maze_example4['f'])
maze_controller(maze_runner)
print maze_runner.found()   # True
```

## приклад лабіринту №5 (складний):

```
maze_example5 = {
    'm': [
        [0,0,0,1,1,0,1,1,0,0,0],
        [0,1,0,0,0,0,0,0,0,1,0],
        [0,1,0,1,1,1,1,1,0,1,0],
        [0,0,0,1,0,0,0,1,0,0,0],
        [0,0,1,1,0,0,0,1,1,0,0],
        [0,0,1,0,0,0,0,0,1,0,0],
        [0,0,1,0,1,0,1,0,1,0,0],
        [0,0,1,0,0,0,0,0,1,0,0],
        [0,0,1,1,1,0,1,1,1,0,0],
        [0,0,0,0,0,0,0,0,0,0,0],
        [0,0,1,0,1,0,1,0,1,0,0],
    ],
    's': (0,5),
    'f': (4,5)
}
maze_runner = MazeRunner(maze_example5['m'], maze_example5['s'],
maze_example5['f'])
maze_controller(maze_runner)
print maze_runner.found()   # True
```