СОРТИРОВКА (sortic.py и check_sortic.py)

• ***136 кг*** *пригодных для переработки материалов – бумагу, стекло, пластик, металл и другие – ежегодно выбрасывает в среднем каждый житель Москвы.*

• *Вплоть до **XX** века наиболее активно подвергавшимся переработке материалом были кости животных. Из полученного сырья производили желатин, клей, пуговицы и даже бумагу.*

• *Около **500 лет** потребуется для полного разложения в природе одноразовых подгузников, гигиенических прокладок, платков и других изделий из нетканых материалов в природе.*

• ***27 тысяч** деревьев ежегодно превращаются в туалетную бумагу. Этот продукт также может быть изготовлен из вторсырья.*

• ***В XXI веке 95%** информации по-прежнему хранится на бумажных носителях, при этом подавляющее большинство документов просматривается лишь один раз.*

• ***9 из 10** произведенных стеклянных бутылок в Финляндии перерабатываются или используются повторно.*

Проект Sortic - это очень простой и высокоэффективный проект алгоритма: данные должны быть отсортированы. В вашем распоряжении набор значений int (integer) 2 стека (стек – условное название, это массив или список) и набор инструкций для управления обоими стеками.

Ваша цель?

Напишите 2 программы на языке Python:

• Первая называется **sortic**, который вычисляет и отображает на стандартном выходе наименьшую программу с использованием языка инструкций Sortic, сортирующего полученные целочисленные аргументы.

Легко? Ну, это мы еще посмотрим…

• Вторая **Check_sortic**, именованная проверка, которая принимает целочисленные аргументы и считывает инструкции на стандартном выходе. После прочтения Check_sortic выполняет их и отображает **OK**, если целые числа отсортированы. В противном случае она будет отображать **KO**.

Написание алгоритма сортировки всегда является очень важным шагом в жизни программиста, потому что это часто первая встреча с понятием сложности. Алгоритмы сортировки и их сложности являются частью классических вопросов, обсуждаемых во время собеседований. Вероятно, сейчас самое подходящее время взглянуть на эти концепции, потому что вам придется столкнуться с ними в какой-то момент. Учебными целями этого проекта являются корректность использование языка Python и

использование базовых алгоритмов. Особенно учитывая сложность этих базовых алгоритмов. Сортировка значений проста. Сортировать их максимально быстрым способом не так просто, тем более что от одной конфигурации целых чисел к другой наиболее эффективный алгоритм сортировки может отличаться.

- В проекте вы можете использовать только целочисленные массивы (вектора, списки).
- В проекте вы можете использовать только собственные функции.
- Вы можете организовывать и называть свои файлы по своему усмотрению, хотя вам необходимо соблюдать некоторые требования, перечисленные ниже.
  • Первый исполняемый файл должен быть назван **sortic.py**, а второй **check_sortic.py**
  • У вас могут быть и другие файлы, но они все должны содержать не больше 5 функций. Каждая функция не превышает 25 строк.
- Если вы умны, вы будете использовать уже созданные ранее Вами функции.
- Ваш проект должен быть написан на языке Python в соответствии с нормой. (соблюдайте отступы во время вложенности)
- Вы должны обращаться с ошибками деликатно. Ни в коем случае ваша программа не может выйти из системы неожиданным образом (выход за границу массива, ошибка сегментации, ошибка шины и т. д.).

ПОЯСНЕНИЯ:

- Игра состоит из 2 стеков с именами **a** и **b**.
- a содержит случайное число положительных или отрицательных чисел без каких-либо дубликатов.
- b пусто (данный стэк вы заводите сами, какой он должен быть тоже придумываете сами, если Вам нужны какие-то еще стэки их тоже заводите).
- Цель состоит в том, чтобы отсортировать номера в порядке возрастания в стек a.
- Для этого в вашем распоряжении имеются следующие операции:
  sa: меняет местами первые 2 элемента на вершине стека a. Ничего не делать, если есть только один или нет элементов).
  sb: поменять местами первые 2 элемента в верхней части стека b. Ничего не делать, если есть только один или нет элементов).
  ss: sa и sb одновременно.
  pa: взять первый элемент в верхней части b и поместите его в верхнюю часть a. ничего не делайте, если b пуст.
  pb: взять первый элемент в верхней части a и поместите его в верхнюю часть b. ничего не делайте, если a пуст.
  ra: сдвиг всех элементов стека a на 1 вверх. Первый элемент становится последним.
  rb: сдвиг вверх всех элементов стека b на 1. Первый элемент становится последним.
  rr: ra и rb одновременно.
  rra: сдвиг всех элементов стека a на 1 вниз
  rrb: сдвиг всех элементов стека b на 1 вниз
  rrr: одновременно выполнить rra и rrb

```
Init a and b:
2
1
3
6
5
8

- -
a b
```

```
Exec sa:
1
2
3
6
5
8

- -
a b
```

```
Exec pb pb pb:
6 3
5 2
8 1

- -
a b
```

```
Exec ra rb (equiv. to rr):
5 2
8 1
6 3

- -
a b
```

```
Exec rra rrb (equiv. to rrr):
6 3
5 2
8 1

- -
a b
```

```
Exec pa pa pa:
1
2
3
5
6
8

- -
```

Пример:
Данные которые подает пользователь после запуска sortic:
2
1
3
6
5
8
!
Ответ программы:
sa
pb
pb
pb
ra
rb
rra
rrb
pa
pa
pa
Ответ может отличаться от оригинала (зависит от вашего алгоритма)

Пример:
Данные которые подает пользователь после запуска check_sortic:
2
1
3
6
5
8
!
sa
pb
pb
pb
ra
rb
rra
rrb
pa
pa
pa
*
Ответ программы:
OK

Пример:
Данные которые подает пользователь после запуска check_sortic:
2
1
3
6
5
8
!
sa
pb
pb
pb
*
Ответ:
КО

Для анализа: Числа всегда вводятся различные (не превышающие 1000)! В конце ввода чисел всегда идет знак «!». Команды всегда вводятся только существующие, в конце ввода команд всегда идет знак «*».

Бонусы (оцениваются отдельно, при условии полной работы основной части. Один бонус – одна пятерка на команду):

1. Сделайте подсветку одной любой из команды (здесь можно использовать сторонние функции не написанные Вами)
2. Чтение из файла данных и запись результата в файл
3. Визуализация того, что в стэках лежит в данный момент (после работы каждой команды)
4. Ввод аргументов одной строкой 2 1 3 6 5 8
5. Передача аргументов в виде строки аргументов при запуске программы: Пример: sortic.py “2 1 3 6 5 8”

   Сортировка, выполненная за меньшее количество команд среди всех участников, будет вознаграждена пятеркой.

   Твоя команда всегда может спросить или обсудить что-то с преподавателем (даже алгоритм)

   Все коды проверяются на плагиат – не попадись…