**Министерство образования и науки Российской Федерации**

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ,   
МЕХАНИКИ И ОПТИКИ**

Факультет программной инженерии и компьютерной техники  
Направление подготовки 09.03.04 Программная инженерия

Дисциплина «Алгоритмы и структуры данных»

**ОТЧЁТ**

по лабораторной работе №4 (Week 4 Openedu)

Студент Дунаев Алексей Игоревич

Группа P3217

Преподаватель Муромцев Дмитрий Ильич

Санкт-Петербург

2019 г.

# Задача 1 Стек

|  |  |
| --- | --- |
| Имя входного файла: | input.txt |
| Имя выходного файла: | output.txt |
| Ограничение по времени: | 2 секунды |
| Ограничение по памяти: | 256 мегабайт |

Реализуйте работу стека. Для каждой операции изъятия элемента выведите ее результат.

На вход программе подаются строки, содержащие команды. Каждая строка содержит одну команду. Команда — это либо "+ N”", либо "−". Команда "+ " означает добавление в стек числа , по модулю не превышающего . Команда "−" означает изъятие элемента из стека. Гарантируется, что не происходит извлечения из пустого стека. Гарантируется, что размер стека в процессе выполнения команд не превысит элементов.

#### Формат входного файла

В первой строке входного файла содержится ( ) — число команд. Каждая последующая строка исходного файла содержит ровно одну команду.

#### Формат выходного файла

Выведите числа, которые удаляются из стека с помощью команды "−", по одному в каждой строке. Числа нужно выводить в том порядке, в котором они были извлечены из стека. Гарантируется, что изъятий из пустого стека не производится.

#### Пример

|  |  |
| --- | --- |
| input.txt | output.txt |
| 6 + 1 + 10 - + 2 + 1234 - | 10 1234 |

## Исходный код к задаче 1

#include <iostream>

#include <string>

using namespace std;

/\*\*/

#include "edx-io.hpp"

#define cin io

#define cout io

/\*\*/

int main() {

int n;

long long t;

char cmd;

long long \*stack = new long long[1000000];

long long index = 0;

cin >> n;

for (int i = 0; i < n; i++) {

cin >> cmd;

if (cmd == '+') {

cin >> stack[index++];

}

else {

cout << stack[--index] << '\n';

}

}

return 0;

}

## Бенчмарк к задаче 1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№ теста** | **Результат** | **Время, с** | **Память** | **Размер входного файла** | **Размер выходного файла** |
| Max |  | 0.671 | 98164736 | 13389454 | 5693807 |
| 1 | OK | 0.031 | 11350016 | 33 | 10 |
| 2 | OK | 0.031 | 11366400 | 11 | 3 |
| 3 | OK | 0.046 | 11354112 | 19 | 6 |
| 4 | OK | 0.031 | 11374592 | 19 | 6 |
| 5 | OK | 0.031 | 11362304 | 19 | 6 |
| 6 | OK | 0.031 | 11358208 | 96 | 45 |
| 7 | OK | 0.078 | 11366400 | 85 | 56 |
| 8 | OK | 0.046 | 11358208 | 129 | 11 |
| 9 | OK | 0.031 | 11366400 | 131 | 12 |
| 10 | OK | 0.031 | 11436032 | 859 | 540 |
| 11 | OK | 0.031 | 11370496 | 828 | 573 |
| 12 | OK | 0.031 | 11382784 | 1340 | 11 |
| 13 | OK | 0.031 | 11247616 | 1325 | 12 |
| 14 | OK | 0.015 | 11419648 | 8292 | 5590 |
| 15 | OK | 0.031 | 11431936 | 8212 | 5706 |
| 16 | OK | 0.062 | 11513856 | 13298 | 111 |
| 17 | OK | 0.031 | 11517952 | 13354 | 12 |
| 18 | OK | 0.046 | 14004224 | 82372 | 56548 |
| 19 | OK | 0.046 | 14147584 | 82000 | 56993 |
| 20 | OK | 0.031 | 14671872 | 132796 | 1134 |
| 21 | OK | 0.046 | 14643200 | 133914 | 11 |
| 22 | OK | 0.093 | 27987968 | 819651 | 569557 |
| 23 | OK | 0.093 | 26710016 | 819689 | 569681 |
| 24 | OK | 0.109 | 27983872 | 1328670 | 11294 |
| 25 | OK | 0.093 | 27820032 | 1338543 | 11 |
| 26 | OK | 0.593 | 80633856 | 8196274 | 5693035 |
| 27 | OK | 0.562 | 81190912 | 8193816 | 5693807 |
| 28 | OK | 0.671 | 97771520 | 13286863 | 112020 |
| 29 | OK | 0.656 | 98152448 | 13389454 | 11 |
| 30 | OK | 0.656 | 98164736 | 13388564 | 11 |

# Задача 2 Очередь

|  |  |
| --- | --- |
| Имя входного файла: | input.txt |
| Имя выходного файла: | output.txt |
| Ограничение по времени: | 2 секунды |
| Ограничение по памяти: | 256 мегабайт |

Реализуйте работу очереди. Для каждой операции изъятия элемента выведите ее результат.

На вход программе подаются строки, содержащие команды. Каждая строка содержит одну команду. Команда — это либо «+ N», либо «−». Команда «+ N» означает добавление в очередь числа N, по модулю не превышающего . Команда «−» означает изъятие элемента из очереди. Гарантируется, что размер очереди в процессе выполнения команд не превысит элементов.

#### Формат входного файла

В первой строке входного файла содержится ) — число команд. Каждая последующая строка исходного файла содержит ровно одну команду.

#### Формат выходного файла

Выведите числа, которые удаляются из очереди с помощью команды «−», по одному в каждой строке. Числа нужно выводить в том порядке, в котором они были извлечены из очереди. Гарантируется, что извлечения из пустой очереди не производится.

#### Пример

|  |  |
| --- | --- |
| input.txt | output.txt |
| 4 + 1 + 10 - - | 1 10 |

## Исходный код к задаче 2

#include <iostream>

#include <string>

using namespace std;

/\*\*/

#include "edx-io.hpp"

#define cin io

#define cout io

/\*\*/

int main() {

int n;

long long t;

char cmd;

long long \*stack = new long long[1000000]();

long long lindex = 0;

long long rindex = 0;

cin >> n;

for (int i = 0; i < n; i++) {

cin >> cmd;

if (cmd == '+') {

cin >> stack[rindex++];

}

else {

cout << stack[lindex++] << '\n';

}

}

return 0;

}

## Бенчмарк к задаче 2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№ теста** | **Результат** | **Время, с** | **Память** | **Размер входного файла** | **Размер выходного файла** |
| Max |  | 0.687 | 98308096 | 13389454 | 5693807 |
| 1 | OK | 0.046 | 11517952 | 20 | 7 |
| 2 | OK | 0.031 | 11513856 | 11 | 3 |
| 3 | OK | 0.031 | 11501568 | 19 | 6 |
| 4 | OK | 0.031 | 11526144 | 19 | 6 |
| 5 | OK | 0.031 | 11481088 | 96 | 45 |
| 6 | OK | 0.015 | 11571200 | 85 | 56 |
| 7 | OK | 0.031 | 11526144 | 129 | 12 |
| 8 | OK | 0.031 | 11481088 | 131 | 12 |
| 9 | OK | 0.031 | 11509760 | 859 | 538 |
| 10 | OK | 0.062 | 11530240 | 828 | 573 |
| 11 | OK | 0.031 | 11530240 | 1340 | 12 |
| 12 | OK | 0.031 | 11620352 | 1325 | 12 |
| 13 | OK | 0.031 | 11550720 | 8292 | 5589 |
| 14 | OK | 0.015 | 11554816 | 8212 | 5706 |
| 15 | OK | 0.031 | 11591680 | 13298 | 115 |
| 16 | OK | 0.031 | 11636736 | 13354 | 12 |
| 17 | OK | 0.031 | 14065664 | 82372 | 56552 |
| 18 | OK | 0.031 | 14237696 | 82000 | 56993 |
| 19 | OK | 0.046 | 14794752 | 132796 | 1124 |
| 20 | OK | 0.031 | 14807040 | 133914 | 12 |
| 21 | OK | 0.093 | 28143616 | 819651 | 569553 |
| 22 | OK | 0.093 | 26779648 | 819689 | 569681 |
| 23 | OK | 0.109 | 27389952 | 1328670 | 11296 |
| 24 | OK | 0.093 | 27889664 | 1338543 | 12 |
| 25 | OK | 0.625 | 80711680 | 8196274 | 5693025 |
| 26 | OK | 0.609 | 81244160 | 8193816 | 5693807 |
| 27 | OK | 0.671 | 97927168 | 13286863 | 112110 |
| 28 | OK | 0.687 | 98308096 | 13389454 | 10 |
| 29 | OK | 0.687 | 98279424 | 13388564 | 11 |

# Задача 3 Скобочная последовательность

|  |  |
| --- | --- |
| Имя входного файла: | input.txt |
| Имя выходного файла: | output.txt |
| Ограничение по времени: | 2 секунды |
| Ограничение по памяти: | 256 мегабайт |

Последовательность A, состоящую из символов из множества «(», «)», «[» и «]», назовем правильной скобочной последовательностью, если выполняется одно из следующих утверждений:

 A — пустая последовательность;

 первый символ последовательности A — это «(», и в этой последовательности существует такой символ «)», что последовательность можно представить как A=(B)C, где B и C — правильные скобочные последовательности;

 первый символ последовательности A — это «[», и в этой последовательности существует такой символ «]», что последовательность можно представить как A=[B]C , где B и C— правильные скобочные последовательности.

Так, например, последовательности «(())» и «()[]» являются правильными скобочными последовательностями, а последовательности «[)» и «((» таковыми не являются.

Входной файл содержит несколько строк, каждая из которых содержит последовательность символов «(», «)», «[» и «]». Для каждой из этих строк выясните, является ли она правильной скобочной последовательностью.

#### Формат входного файла

Первая строка входного файла содержит число N( - число скобочных последовательностей, которые необходимо проверить. Каждая из следующих N строк содержит скобочную последовательность длиной от 1 до включительно. В каждой из последовательностей присутствуют только скобки указанных выше видов.

#### Формат выходного файла

Для каждой строки входного файла выведите в выходной файл «YES», если соответствующая последовательность является правильной скобочной последовательностью, или «NO», если не является.

#### Пример

|  |  |
| --- | --- |
| input.txt | output.txt |
| 5 ()() ([]) ([)] ((]] )( | YES YES NO NO NO |

## Исходный код к задаче 3

#include <iostream>

#include <string>

using namespace std;

/\*\*/

#include "edx-io.hpp"

#define cin io

#define cout io

/\*\*/

int check() {

string cmd;

long long \*queue = new long long[10000];

long index = 0;

cin >> cmd;

for (int j = 0; j < cmd.size(); j++) {

switch (cmd[j])

{

case '(': case '[':

queue[index++] = cmd[j];

break;

case ')': case ']':

if (index <= 0 || queue[--index] != (cmd[j] == ')' ? '(' : '[')) {

//cout << index << " " << queue[index] << " " << cmd[j] << " ";

// cout << "NO\n";

// goto end;

return 1;

}

break;

}

}

// cout << (index ? "NO\n" : "YES\n");

// end:;

return index;

}

int main() {

int n;

cin >> n;

for (int i = 0; i < n; i++) {

cout << (check() ? "NO\n" : "YES\n");

}

return 0;

}

## Бенчмарк к задаче 3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№ теста** | **Результат** | **Время, с** | **Память** | **Размер входного файла** | **Размер выходного файла** |
| Max |  | 0.140 | 27168768 | 5000885 | 2133 |
| 1 | OK | 0.031 | 10633216 | 31 | 22 |
| 2 | OK | 0.031 | 10838016 | 15 | 16 |
| 3 | OK | 0.031 | 10711040 | 68 | 66 |
| 4 | OK | 0.031 | 10682368 | 324 | 256 |
| 5 | OK | 0.031 | 10772480 | 1541 | 1032 |
| 6 | OK | 0.031 | 10764288 | 5880 | 2128 |
| 7 | OK | 0.031 | 10989568 | 50867 | 2129 |
| 8 | OK | 0.093 | 13393920 | 500879 | 2110 |
| 9 | OK | 0.125 | 27164672 | 5000884 | 2120 |
| 10 | OK | 0.140 | 27168768 | 5000885 | 2133 |

# Задача 4 Очередь с минимумом

|  |  |
| --- | --- |
| Имя входного файла: | input.txt |
| Имя выходного файла: | output.txt |
| Ограничение по времени: | 2 секунды |
| Ограничение по памяти: | 256 мегабайт |

Реализуйте работу очереди. В дополнение к стандартным операциям очереди, необходимо также отвечать на запрос о минимальном элементе из тех, которые сейчас находится в очереди. Для каждой операции запроса минимального элемента выведите ее результат.

На вход программе подаются строки, содержащие команды. Каждая строка содержит одну команду. Команда — это либо «+ N», либо «−», либо «?». Команда «+ N» означает добавление в очередь числа N, по модулю не превышающего . Команда «−» означает изъятие элемента из очереди. Команда «?» означает запрос на поиск минимального элемента в очереди.

#### Формат входного файла

В первой строке содержится )— число команд. В последующих строках содержатся команды, по одной в каждой строке.

#### Формат выходного файла

Для каждой операции поиска минимума в очереди выведите её результат. Результаты должны быть выведены в том порядке, в котором эти операции встречаются во входном файле. Гарантируется, что операций извлечения или поиска минимума для пустой очереди не производится.

#### Пример

|  |  |
| --- | --- |
| input.txt | output.txt |
| 7 + 1 ? + 10 ? - ? - | 1 1 10 |

## Исходный код к задаче 4

#include <iostream>

#include <string>

#include <queue>

#include <set>

using namespace std;

/\*\*/

#include "edx-io.hpp"

#define cin io

#define cout io

/\*\*/

int main() {

int n;

long long t;

char cmd;

queue<long long> q;

multiset<long long> s;

cin >> n;

for (int i = 0; i < n; i++) {

cin >> cmd;

switch (cmd)

{

case '+':

cin >> t;

q.push(t);

s.insert(t);

break;

case '-':

s.erase(s.find(q.front()));

q.pop();

break;

case '?':

cout << \*s.begin() << "\n";

break;

}

}

return 0;

}

## Бенчмарк к задаче 4

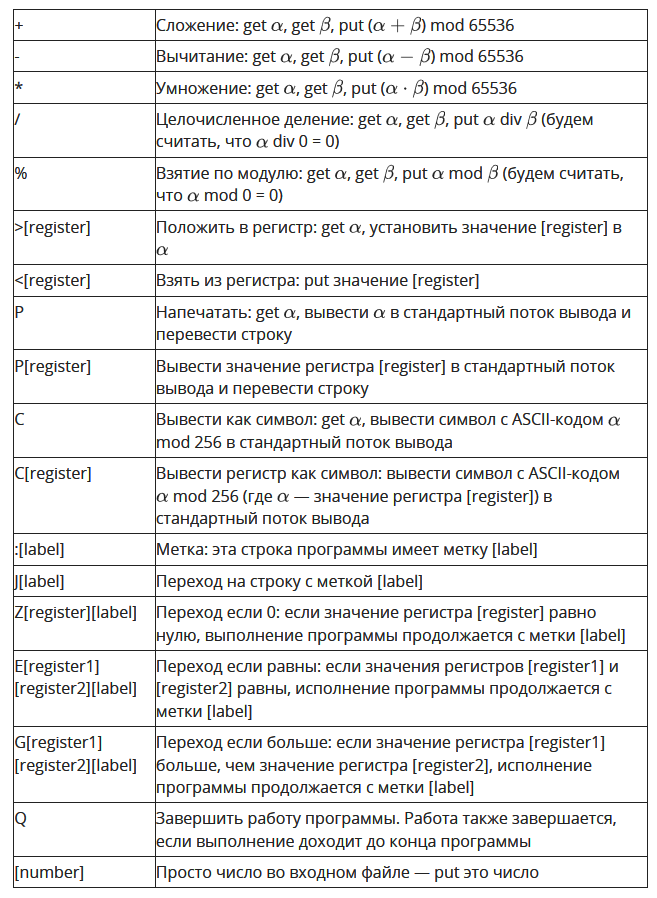
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№ теста** | **Результат** | **Время, с** | **Память** | **Размер входного файла** | **Размер выходного файла** |
| Max |  | 1.203 | 145141760 | 13389342 | 4002151 |
| 1 | OK | 0.031 | 11444224 | 29 | 10 |
| 2 | OK | 0.031 | 11386880 | 11 | 3 |
| 3 | OK | 0.031 | 11431936 | 22 | 6 |
| 4 | OK | 0.031 | 11415552 | 22 | 6 |
| 5 | OK | 0.046 | 11448320 | 36 | 9 |
| 6 | OK | 0.015 | 11472896 | 48 | 12 |
| 7 | OK | 0.031 | 11448320 | 76 | 35 |
| 8 | OK | 0.031 | 11427840 | 129 | 12 |
| 9 | OK | 0.031 | 11440128 | 67 | 48 |
| 10 | OK | 0.046 | 11440128 | 44 | 9 |
| 11 | OK | 0.031 | 11481088 | 45 | 9 |
| 12 | OK | 0.031 | 11464704 | 44 | 9 |
| 13 | OK | 0.031 | 11378688 | 45 | 9 |
| 14 | OK | 0.031 | 11411456 | 721 | 384 |
| 15 | OK | 0.031 | 11390976 | 1340 | 12 |
| 16 | OK | 0.031 | 11444224 | 640 | 407 |
| 17 | OK | 0.031 | 11444224 | 445 | 90 |
| 18 | OK | 0.046 | 11427840 | 456 | 100 |
| 19 | OK | 0.031 | 11522048 | 445 | 90 |
| 20 | OK | 0.031 | 11464704 | 456 | 100 |
| 21 | OK | 0.031 | 11661312 | 6616 | 3812 |
| 22 | OK | 0.031 | 11776000 | 13389 | 12 |
| 23 | OK | 0.031 | 11640832 | 6461 | 4008 |
| 24 | OK | 0.031 | 11599872 | 4896 | 1140 |
| 25 | OK | 0.046 | 11628544 | 5007 | 1250 |
| 26 | OK | 0.046 | 11616256 | 4896 | 1140 |
| 27 | OK | 0.062 | 11603968 | 5007 | 1250 |
| 28 | OK | 0.046 | 13471744 | 64907 | 39589 |
| 29 | OK | 0.031 | 15343616 | 133814 | 12 |
| 30 | OK | 0.062 | 13516800 | 64675 | 39996 |
| 31 | OK | 0.031 | 14102528 | 53897 | 13890 |
| 32 | OK | 0.031 | 13545472 | 55008 | 15000 |
| 33 | OK | 0.031 | 14131200 | 53897 | 13890 |
| 34 | OK | 0.046 | 13598720 | 55008 | 15000 |
| 35 | OK | 0.093 | 28278784 | 645271 | 404305 |
| 36 | OK | 0.109 | 28430336 | 1338956 | 12 |
| 37 | OK | 0.078 | 28520448 | 646300 | 400008 |
| 38 | OK | 0.109 | 28590080 | 588898 | 163890 |
| 39 | OK | 0.078 | 28454912 | 600009 | 175000 |
| 40 | OK | 0.109 | 28585984 | 588898 | 163890 |
| 41 | OK | 0.093 | 28475392 | 600009 | 175000 |
| 42 | OK | 0.531 | 80834560 | 6465010 | 4002151 |
| 43 | OK | 0.734 | 91480064 | 13389342 | 12 |
| 44 | OK | 0.531 | 80977920 | 6462989 | 4000004 |
| 45 | OK | 0.734 | 107687936 | 6388899 | 1888890 |
| 46 | OK | 0.531 | 80949248 | 6500010 | 2000000 |
| 47 | OK | 0.781 | 108339200 | 6388899 | 1888890 |
| 48 | OK | 0.531 | 80965632 | 6500010 | 2000000 |
| 49 | OK | 0.734 | 91447296 | 13388086 | 12 |
| 50 | OK | 0.031 | 11403264 | 55 | 16 |
| 51 | OK | 0.031 | 11448320 | 705 | 225 |
| 52 | OK | 0.031 | 11649024 | 6506 | 2000 |
| 53 | OK | 0.031 | 14249984 | 65007 | 20000 |
| 54 | OK | 0.125 | 28815360 | 650008 | 200000 |
| 55 | OK | 0.812 | 107147264 | 6675213 | 2000000 |
| 56 | OK | 0.031 | 11423744 | 117 | 12 |
| 57 | OK | 0.031 | 11427840 | 1327 | 12 |
| 58 | OK | 0.046 | 11763712 | 13417 | 12 |
| 59 | OK | 0.046 | 15310848 | 133845 | 12 |
| 60 | OK | 0.156 | 30552064 | 1339319 | 12 |
| 61 | OK | 1.203 | 145141760 | 13388955 | 12 |

# Задача 5 Quack

|  |  |
| --- | --- |
| Имя входного файла: | input.txt |
| Имя выходного файла: | output.txt |
| Ограничение по времени: | 2 секунды |
| Ограничение по памяти: | 256 мегабайт |

Язык Quack — забавный язык, который фигурирует в одной из задач с [Internet Problem Solving Contest](http://ipsc.ksp.sk/). В этой задаче вам требуется написать интерпретатор языка Quack.

Виртуальная машина, на которой исполняется программа на языке Quack, имеет внутри себя очередь, содержащую целые числа по модулю 65536 (то есть, числа от 0 до 65535, соответствующие беззнаковому 16-битному целому типу). Слово get в описании операций означает извлечение из очереди, put — добавление в очередь. Кроме того, у виртуальной машины есть 26 регистров, которые обозначаются буквами от 'a' до 'z'. Изначально все регистры хранят нулевое значение. В языке Quack существуют следующие команды (далее под a и b подразумеваются некие абстрактные временные переменные):



#### Формат входного файла

Входной файл содержит синтаксически корректную программу на языке Quack. Известно, что программа завершает работу не более чем за

шагов. Программа содержит не менее одной и не более

инструкций. **Метки имеют длину от 1 до 10 и состоят из цифр и латинских букв.**

#### Формат выходного файла

Выведите содержимое стандартного потока вывода виртуальной машины в выходной файл.

#### Примеры

|  |  |
| --- | --- |
| input.txt | output.txt |
| 100 0 :start >a Zaend <a <a 1 + - >b <b Jstart :end P | 5050 |

Второй пример подразумевает UNIX-переводы строки в ответе (один символ с кодом 10).

|  |  |
| --- | --- |
| input.txt | output.txt |
| 58 49 10 62 97 10 80 97 10 90 97 50 10 60 97 10 74 49 10 58 50 10 48 10 58 51 10 62 97 10 90 97 52 10 67 97 10 74 51 10 58 52 10 0 :1 >a Pa Za2 <a J1 :2 0 :3 >a Za4 Ca J3 :4 | 58 49 10 62 97 10 80 97 10 90 97 50 10 60 97 10 74 49 10 58 50 10 48 10 58 51 10 62 97 10 90 97 52 10 67 97 10 74 51 10 58 52 10 0 :1 >a Pa Za2 <a J1 :2 0 :3 >a Za4 Ca J3 :4 |

## Исходный код к задаче 5

import queue

from edx\_io import edx\_io

registers = {}

queue = queue.Queue()

labels = {}

p = 0

q = False

def put(v):

queue.put(v % 65536, False)

def get():

return queue.get(False)

def jump(lbl):

global p

p = labels[lbl]

def do\_command(cmd, io):

c = cmd[0]

if c == '+':

put(get() + get())

elif c == '-':

a = get()

put(a - get())

elif c == '\*':

put(get() \* get())

elif c == '/':

a = get()

b = get()

put(0 if b == 0 else a // b)

elif c == '%':

a = get()

b = get()

put(0 if b == 0 else a % b)

elif c == '>':

registers[cmd[1]] = get()

elif c == '<':

put(registers[cmd[1]])

elif c == 'P':

if len(cmd) == 1:

io.writeln(get())

else:

io.writeln(registers[cmd[1]])

elif c == 'C':

if len(cmd) == 1:

io.write(chr(get() % 256))

else:

io.write(chr(registers[cmd[1]] % 256))

elif c == ':':

labels[cmd[1:]] = p

elif c == 'J':

jump(cmd[1:])

elif c == 'Z':

if registers[cmd[1]] == 0:

jump(cmd[2:])

elif c == 'E':

if registers[cmd[1]] == registers[cmd[2]]:

jump(cmd[3:])

elif c == 'G':

if registers[cmd[1]] > registers[cmd[2]]:

jump(cmd[3:])

elif c == 'Q':

global q

q = True

else:

put(int(cmd))

with edx\_io() as io:

tokens = [t.decode('ascii') for t in io.all\_tokens()]

labels = {

label[1:]: i

for i, label in enumerate(tokens)

if label[0] == ':'

}

registers = {

chr(letter): 0 for letter in range(ord('a'), ord('z') + 1)

}

lt = len(tokens)

while p < lt and not q:

do\_command(tokens[p], io)

p += 1

## Бенчмарк к задаче 5

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№ теста** | **Результат** | **Время, с** | **Память** | **Размер входного файла** | **Размер выходного файла** |
| Max |  | 0.109 | 24506368 | 1349803 | 250850 |
| 1 | OK | 0.015 | 11001856 | 69 | 6 |
| 2 | OK | 0.031 | 11005952 | 232 | 218 |
| 3 | OK | 0.031 | 10915840 | 3 | 0 |
| 4 | OK | 0.015 | 10944512 | 100 | 19 |
| 5 | OK | 0.031 | 11800576 | 56 | 58890 |
| 6 | OK | 0.046 | 11366400 | 67 | 30000 |
| 7 | OK | 0.031 | 11431936 | 67 | 30000 |
| 8 | OK | 0.062 | 11423744 | 55 | 30000 |
| 9 | OK | 0.046 | 10952704 | 461 | 60 |
| 10 | OK | 0.031 | 11186176 | 11235 | 21 |
| 11 | OK | 0.015 | 11395072 | 23748 | 42 |
| 12 | OK | 0.031 | 12091392 | 66906 | 8905 |
| 13 | OK | 0.015 | 11055104 | 7332 | 954 |
| 14 | OK | 0.031 | 11149312 | 4611 | 602 |
| 15 | OK | 0.031 | 11591680 | 37968 | 5424 |
| 16 | OK | 0.031 | 10956800 | 14 | 2 |
| 17 | OK | 0.015 | 10944512 | 70 | 10 |
| 18 | OK | 0.031 | 11075584 | 350 | 50 |
| 19 | OK | 0.031 | 11018240 | 1750 | 250 |
| 20 | OK | 0.031 | 11145216 | 8750 | 1250 |
| 21 | OK | 0.031 | 11706368 | 43750 | 6250 |
| 22 | OK | 0.046 | 14839808 | 218750 | 31250 |
| 23 | OK | 0.031 | 11567104 | 34606 | 4721 |
| 24 | OK | 0.078 | 18477056 | 683180 | 7 |
| 25 | OK | 0.046 | 18472960 | 683102 | 0 |
| 26 | OK | 0.062 | 24506368 | 1349803 | 0 |
| 27 | OK | 0.078 | 19132416 | 491572 | 247791 |
| 28 | OK | 0.062 | 19095552 | 491488 | 249618 |
| 29 | OK | 0.062 | 19144704 | 491600 | 249600 |
| 30 | OK | 0.062 | 19144704 | 491502 | 250850 |
| 31 | OK | 0.062 | 19103744 | 491416 | 249477 |
| 32 | OK | 0.078 | 19111936 | 491520 | 250262 |
| 33 | OK | 0.078 | 19103744 | 491317 | 246859 |
| 34 | OK | 0.046 | 19165184 | 491514 | 248199 |
| 35 | OK | 0.109 | 19152896 | 491557 | 249601 |

## **Постфиксная запись**

 Добавить страницу в мои закладки

### Постфиксная запись

0 возможных балла (не оценивается)

|  |  |
| --- | --- |
| Имя входного файла: | input.txt |
| Имя выходного файла: | output.txt |
| Ограничение по времени: | 2 секунды |
| Ограничение по памяти: | 256 мегабайт |

В постфиксной записи (или обратной польской записи) операция записывается после двух операндов. Например, сумма двух чисел A и B записывается как A B +. Запись B C + D ∗ обозначает привычное нам (B + C) ∗ D, а запись A B C + D ∗ + означает A + (B + C) ∗ D. Достоинство постфиксной записи в том, что она не требует скобок и дополнительных соглашений о приоритете операторов для своего чтения.

Дано выражение в обратной польской записи. Определите его значение.

#### Формат входного файла

В первой строке входного файла дано число  () - число элементов выражения. Во второй строке содержится выражение в постфиксной записи, состоящее из  элементов. В выражении могут содержаться неотрицательные однозначные числа и операции +, −, ∗. Каждые два соседних элемента выражения разделены ровно одним пробелом.

#### Формат выходного файла

Необходимо вывести значение записанного выражения. Гарантируется, что результат выражения, а также результаты всех промежуточных вычислений, по модулю будут меньше, чем .

#### Пример

|  |  |
| --- | --- |
| input.txt | output.txt |
| 7 8 9 + 1 7 - \* | -102 |

### Искходный код

### from edx\_io import edx\_io

### qu = []

### with edx\_io() as io:

### n = io.next\_int()

### for x in range(0,n):

### sign = io.next\_token()

### if sign == b'+':

### qu.append(qu.pop()+qu.pop())

### elif sign == b'-':

### qu.append(-qu.pop()+qu.pop())

### elif sign == b'\*':

### qu.append(qu.pop()\*qu.pop())

### else:

### qu.append(int(sign))

### io.writeln(qu.pop())

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № **теста** | **Результат** | **Время, с** | **Память** | **Размер входного файла** | **Размер выходного файла** |
| Max |  | 1.203 | 32030720 | 2000007 | 12 |
| 1 | OK | 0.062 | 11038720 | 18 | 5 |
| 2 | OK | 0.062 | 10989568 | 6 | 2 |
| 3 | OK | 0.046 | 10977280 | 10 | 2 |
| 4 | OK | 0.062 | 11001856 | 39 | 10 |
| 5 | OK | 0.062 | 10981376 | 39 | 4 |
| 6 | OK | 0.078 | 11001856 | 10 | 3 |
| 7 | OK | 0.046 | 10985472 | 10 | 3 |
| 8 | OK | 0.062 | 11014144 | 10 | 3 |
| 9 | OK | 0.046 | 11022336 | 39 | 7 |
| 10 | OK | 0.046 | 10948608 | 63 | 5 |
| 11 | OK | 0.046 | 11005952 | 10 | 2 |
| 12 | OK | 0.062 | 10989568 | 2004 | 4 |
| 13 | OK | 0.046 | 11005952 | 2004 | 3 |
| 14 | OK | 0.062 | 11014144 | 1924 | 8 |
| 15 | OK | 0.062 | 11067392 | 1992 | 12 |
| 16 | OK | 0.062 | 10997760 | 1992 | 11 |
| 17 | OK | 0.109 | 10969088 | 1996 | 2 |
| 18 | OK | 0.062 | 11026432 | 1984 | 7 |
| 19 | OK | 0.062 | 11014144 | 2004 | 3 |
| 20 | OK | 0.046 | 10993664 | 1984 | 7 |
| 21 | OK | 0.062 | 11059200 | 2004 | 3 |
| 22 | OK | 0.062 | 11116544 | 20005 | 5 |
| 23 | OK | 0.062 | 11186176 | 20005 | 3 |
| 24 | OK | 0.078 | 11173888 | 19461 | 8 |
| 25 | OK | 0.062 | 11141120 | 19977 | 12 |
| 26 | OK | 0.062 | 11116544 | 19977 | 11 |
| 27 | OK | 0.062 | 11161600 | 19981 | 2 |
| 28 | OK | 0.062 | 11137024 | 19985 | 6 |
| 29 | OK | 0.078 | 11202560 | 20005 | 5 |
| 30 | OK | 0.078 | 11161600 | 19985 | 8 |
| 31 | OK | 0.062 | 11194368 | 20005 | 5 |
| 32 | OK | 0.156 | 12713984 | 200006 | 6 |
| 33 | OK | 0.171 | 13778944 | 200006 | 2 |
| 34 | OK | 0.171 | 14008320 | 196614 | 8 |
| 35 | OK | 0.171 | 13307904 | 196618 | 12 |
| 36 | OK | 0.171 | 13762560 | 196618 | 11 |
| 37 | OK | 0.187 | 13217792 | 196622 | 2 |
| 38 | OK | 0.171 | 13275136 | 199986 | 6 |
| 39 | OK | 0.156 | 13279232 | 200006 | 6 |
| 40 | OK | 0.171 | 13815808 | 199986 | 8 |
| 41 | OK | 0.156 | 13316096 | 200006 | 4 |
| 42 | OK | 1.140 | 30064640 | 2000007 | 7 |
| 43 | OK | 1.140 | 31997952 | 2000007 | 3 |
| 44 | OK | 0.968 | 26959872 | 1572871 | 8 |
| 45 | OK | 1.203 | 29999104 | 1966091 | 12 |
| 46 | OK | 1.156 | 30007296 | 1966091 | 11 |
| 47 | OK | 1.156 | 30142464 | 1966095 | 2 |
| 48 | OK | 1.140 | 30121984 | 1999987 | 8 |
| 49 | OK | 1.156 | 32030720 | 2000007 | 4 |
| 50 | OK | 1.156 | 30085120 | 1999987 | 7 |
| 51 | OK | 1.140 | 32010240 | 2000007 | 5 |