Санкт-Петербургский национальный исследовательский университет
информационных технологий, механики и оптики

Факультет Программной инженерии и компьютерной техники
Кафедра информатики и прикладной математики

Дисциплина «Алгоритмы и структуры данных»
Лабораторная работа №4

Выполнил:
Луговских Савелий Михайлович
Группа P3218

Преподаватель:
Муромцев Дмитрий Ильич

Санкт-Петербург, 2019 г.

1.Стек

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 "+ N" означает добавление в стек числа N, по модулю не превышающего 109. Команда "-" означает изъятие элемента из стека. Гарантируется, что не происходит извлечения из пустого стека. Гарантируется, что размер стека в процессе выполнения команд не превысит 106 элементов.

Формат входного файла

В первой строке входного файла содержится M (1≤M≤106) — 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.

Исходный код(C#)

```
namespace
Stack
            {
                using System.IO;

                class Program
                {
                    static void Main()
                    {
                        string[] inputStack = File.ReadAllLines("input.txt");
                        StreamWriter output = new StreamWriter("output.txt");
                        string[] bufStack = new string[inputStack.Length];
                        int stackLength = bufStack.Length - 1;
                        for (int i = 1; i < inputStack.Length; i++)
                        {
                            if (inputStack[i][0] == '+')
                            {
                                bufStack[stackLength] = inputStack[i].Substring(2);
                                --stackLength;
                            }
                            else
                            {
                                output.WriteLine(bufStack[stackLength + 1]);
```

```
                    ++stackLength;
                }
            }

            output.Close();
        }
    }
}
```

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.453	139083776	13389454	5693807
1	OK	0.031	9809920	33	10
2	OK	0.015	9838592	11	3
3	OK	0.031	9859072	19	6
4	OK	0.031	9805824	19	6
5	OK	0.031	9793536	19	6
6	OK	0.015	9793536	96	45
7	OK	0.015	9859072	85	56
8	OK	0.031	9920512	129	11
9	OK	0.031	9814016	131	12
10	OK	0.046	9789440	859	540
11	OK	0.031	9830400	828	573
12	OK	0.031	9801728	1340	11
13	OK	0.015	9850880	1325	12
14	OK	0.031	9965568	8292	5590
15	OK	0.031	10006528	8212	5706
16	OK	0.015	9916416	13298	111
17	OK	0.015	9928704	13354	12
18	OK	0.031	11284480	82372	56548
19	OK	0.031	11358208	82000	56993

2.Очередь

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 109. Команда «-» означает изъятие элемента из очереди. Гарантируется, что размер очереди в процессе выполнения команд не превысит 106 элементов.

Формат входного файла

В первой строке содержится M ($1 \le M \le 106$) — число команд. В последующих строках содержатся команды, по одной в каждой строке.

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.

Исходный код (C#)

```
namespace
Queue
            {
                using System.IO;

                class Program
                {
                    static void Main()
                    {
                        string[] inputQueue = File.ReadAllLines("input.txt");
                        StreamWriter output = new StreamWriter("output.txt");
                        string[] bufQueue = new string[inputQueue.Length];
                        int queueLength = 0;
                        int queuePosition = 0;
                        for (int i = 1; i < inputQueue.Length; i++)
                        {
                            if (inputQueue[i][0] == '+')
                            {
                                bufQueue[queueLength] = inputQueue[i].Substring(2);
                                queueLength++;
                            }
                            else
                            {
                                output.WriteLine(bufQueue[queuePosition]);
                                queuePosition++;
                            }
                        }

                        output.Close();
                    }
                }
            }
```

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.515	139001856	13389454	5693807
1	OK	0.015	9912320	20	7
2	OK	0.015	9838592	11	3
3	OK	0.015	9883648	19	6
4	OK	0.015	9846784	19	6
5	OK	0.031	9867264	96	45
6	OK	0.015	9863168	85	56
7	OK	0.031	9846784	129	12
8	OK	0.015	9842688	131	12
9	OK	0.031	9846784	859	538
10	OK	0.015	9834496	828	573
11	OK	0.031	9834496	1340	12
12	OK	0.031	9924608	1325	12
13	OK	0.015	9936896	8292	5589
14	OK	0.015	9986048	8212	5706
15	OK	0.031	9969664	13298	115
16	OK	0.031	9990144	13354	12
17	OK	0.031	11321344	82372	56552
18	OK	0.015	11415552	82000	56993
19	OK	0.031	11513856	132796	1124
20	OK	0.015	11452416	133914	12
21	OK	0.109	22786048	819651	569553
22	OK	0.046	22265856	819689	569681
23	OK	0.046	23781376	1328670	11296

3.Скобочная последовательность

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 ($1 \le N \le 500$) - число скобочных последовательностей, которые необходимо проверить. Каждая из следующих N строк содержит скобочную последовательность длиной от 1 до 104 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.

Исходный код (C#)

```
namespace
Bracket_sequence
                  {
                      using System.IO;

                      class Program
                      {
                          static void Main()
                          {
                              byte[] bracketSequence = File.ReadAllBytes("input.txt");
                              byte[] bufferBrackets = new byte[100000];
                              int startPostion;
                              for (startPostion = 0; startPostion < bracketSequence.Length &&
                  bracketSequence[startPostion] != 10; startPostion++);
                              int lengthBuffer = 0;
                              bool isRight = true;
                              StreamWriter output = new StreamWriter("output.txt");
                              for (int i = startPostion + 1; i < bracketSequence.Length; i++)
                              {
                                  switch (bracketSequence[i])
```

```
                    {
                        case 40:
                            {
                                 bufferBrackets[lengthBuffer] = 40;
                                ++lengthBuffer;
                                break;
                            }

                        case 41:
                            {
                                if (lengthBuffer > 0 &&
bufferBrackets[lengthBuffer - 1] == 40)
                                {
                                    --lengthBuffer;
                                }
                                else
                                {
                                    isRight = false;
                                }

                                break;
                            }

                        case 91:
                            {
                                bufferBrackets[lengthBuffer] = 91;
                                ++lengthBuffer;
                                break;
                            }

                        case 93:
                            {
                                if (lengthBuffer > 0 &&
bufferBrackets[lengthBuffer - 1] == 91)
                                {
                                    --lengthBuffer;
                                }
                                else
                                {
                                    isRight = false;
                                }

                                break;
                            }
```

```
                    case 10:
                        {
                            if (isRight && lengthBuffer == 0)
                            {
                                output.WriteLine("YES");
                            }
                            else
                            {
                                output.WriteLine("NO");
                            }

                            lengthBuffer = 0;
                            isRight = true;
                            break;
                        }
                    }
                }

                output.Close();
            }
        }
    }
```

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.078	15261696	5000885	2133
1	OK	0.031	9867264	31	22
2	OK	0.031	9854976	15	16
3	OK	0.031	9850880	68	66
4	OK	0.031	9883648	324	256
5	OK	0.015	9846784	1541	1032
6	OK	0.046	9895936	5880	2128
7	OK	0.031	9887744	50867	2129
8	OK	0.031	10358784	500879	2110
9	OK	0.062	15163392	5000884	2120
10	OK	0.078	15261696	5000885	2133

4.Очередь с минимумом

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 109. Команда «-» означает изъятие элемента из очереди. Команда «?» означает запрос на поиск минимального элемента в очереди.

Формат входного файла

В первой строке содержится M (1≤M≤106) — 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.

Исходный код (C#)

```
namespace
Queue
            {
                using System;
                using System.Collections.Generic;
                using System.IO;

                class Program
                {
                    static void Main()
                    {
                        string[] inputQueue = File.ReadAllLines("input.txt");
                        StreamWriter output = new StreamWriter("output.txt");
                        Stack<(int, int)> stack1 = new Stack<(int, int)>(inputQueue.Length);
                        Stack<(int, int)> stack2 = new Stack<(int, int)>(inputQueue.Length);
                        int newElement;
                        for (int i = 1; i < inputQueue.Length; i++)
                        {
                            switch (inputQueue[i][0])
                            {
                                case '+':
                                    {
                                        newElement = int.Parse(inputQueue[i].Substring(2));
                                        if (stack1.Count == 0 || stack1.Peek().Item2 >=
            newElement)
                                        {
                                            stack1.Push((newElement, newElement));
                                        }
                                        else
```

```
                            {
                                stack1.Push((newElement, stack1.Peek().Item2));
                            }

                            break;
                        }

                    case '-':
                        {
                            if (stack2.Count == 0)
                            {
                                while (stack1.Count > 0)
                                {
                                    newElement = stack1.Pop().Item1;
                                    if (stack2.Count == 0 || stack2.Peek().Item2
>= newElement)
                                    {
                                        stack2.Push((newElement, newElement));
                                    }
                                    else
                                    {
                                        stack2.Push((newElement,
stack2.Peek().Item2));
                                    }
                                }
                            }

                            stack2.Pop();
                            break;
                        }

                    case '?':
                        {
                            if (stack1.Count == 0 || stack2.Count == 0)
                            {
                                output.WriteLine(stack1.Count == 0 ?
stack2.Peek().Item2 : stack1.Peek().Item2);
                            }
                            else
                            {
                                output.WriteLine(Math.Min(stack1.Peek().Item2,
stack2.Peek().Item2));
                            }

                            break;
```

```
                        }
                    }
                }

                output.Close();
            }
        }
    }
```

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.562	105390080	13389342	4002151
1	OK	0.015	10944512	29	10
2	OK	0.031	10948608	11	3
3	OK	0.015	10977280	22	6
4	OK	0.031	10985472	22	6
5	OK	0.015	10940416	36	9
6	OK	0.046	10924032	48	12
7	OK	0.015	10907648	76	35
8	OK	0.031	11010048	129	12
9	OK	0.046	10924032	67	48
10	OK	0.046	10960896	44	9
11	OK	0.031	11001856	45	9
12	OK	0.031	10932224	44	9
13	OK	0.046	10973184	45	9
14	OK	0.015	10964992	721	384
15	OK	0.031	10969088	1340	12
16	OK	0.031	11022336	640	407
17	OK	0.031	10993664	445	90
18	OK	0.031	10960896	456	100
19	OK	0.031	10977280	445	90
20	OK	0.015	11022336	456	100
21	OK	0.031	11055104	6616	3812
22	OK	0.031	11083776	13389	12
23	OK	0.031	11104256	6461	4008

5. Quack

Язык Quack — забавный язык, который фигурирует в одной из задач с Internet Problem Solving Contest. 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 α и β подразумеваются некие абстрактные временные переменные):

+	Сложение: get α, get β, put (α+β) mod 65536
-	Вычитание: get α, get β, put (α−β) mod 65536
*	Умножение: get α, get β, put (α·β) mod 65536
/	Целочисленное деление: get α, get β, put α div β (будем считать, что α div 0 = 0)
%	Взятие по модулю: get α, get β, put α mod β (будем считать, что α mod 0 = 0)
>[register]	Положить в регистр: get α, установить значение [register] в α
<[register]	Взять из регистра: put значение [register]
P	Напечатать: get α, вывести α в стандартный поток вывода и перевести строку
P[register]	Вывести значение регистра [register] в стандартный поток вывода и перевести строку
C	Вывести как символ: get α, вывести символ с ASCII-кодом α mod 256 в стандартный поток вывода
C[register]	Вывести регистр как символ: вывести символ с ASCII-кодом α mod 256 (где α — значение регистра [register]) в стандартный поток вывода
:[label]	Метка: эта строка программы имеет метку [label]
J[label]	Переход на строку с меткой [label]
Z[register][label]	Переход если 0: если значение регистра [register] равно нулю, выполнение программы продолжается с метки [label]
E[register1][register2][label]	Переход если равны: если значения регистров [register1] и [register2] равны, исполнение программы продолжается с метки [label]
G[register1][register2][label]	Переход если больше: если значение регистра [register1] больше, чем значение регистра [register2], исполнение программы продолжается с метки [label]
Q	Завершить работу программы. Работа также завершается, если выполнение доходит до конца программы
[number]	Просто число во входном файле — put это число

Формат входного файла

Входной файл содержит синтаксически корректную программу на языке Quack. Известно, что программа завершает работу не более чем за 105 шагов. Программа содержит не менее одной и не более 105 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.

Исходный код (C#)

```
namespace
Quack
            {
                using System.Collections.Generic;
                using System.IO;

                class Program
                {
                    static void Main()
                    {
                        string[] inputQueue = File.ReadAllLines("input.txt");
                        StreamWriter output = new StreamWriter("output.txt");
                        bool isEnd = false;
                        long[] registers = new long[26];
                        Dictionary<string, int> labels = new Dictionary<string, int>();
                        long[] bufQueue = new long[10000000];
                        int queueLength = 0;
                        int queuePosition = 0;
                        for (int i = 0; i < inputQueue.Length; i++)
                        {
                            if (inputQueue[i][0] == ':')
                            {
                                labels.Add(inputQueue[i].Substring(1), i);
                            }
                        }

                        for (int i = 0; i < inputQueue.Length && !isEnd; i++)
                        {
                            switch (inputQueue[i][0])
                            {
                                case '+':
                                    {
                                        bufQueue[queueLength] = (bufQueue[queuePosition] +
bufQueue[queuePosition + 1]) & 65535;
```

```
                            queuePosition += 2;
                            ++queueLength;
                            break;
                        }

                    case '-':
                        {
                            bufQueue[queueLength] = (bufQueue[queuePosition] -
bufQueue[queuePosition + 1]) & 65535;
                            queuePosition += 2;
                            ++queueLength;
                            break;
                        }

                    case '/':
                        {
                            if (bufQueue[queuePosition + 1] != 0)
                            {
                                bufQueue[queueLength] = bufQueue[queuePosition] /
bufQueue[queuePosition + 1] & 65535;
                            }
                            else
                            {
                                bufQueue[queueLength] = 0;
                            }

                            queuePosition += 2;
                            ++queueLength;
                            break;
                        }

                    case '%':
                        {
                            if (bufQueue[queuePosition + 1] != 0)
                            {
                                bufQueue[queueLength] = bufQueue[queuePosition] %
bufQueue[queuePosition + 1] & 65535;
                            }
                            else
                            {
                                bufQueue[queueLength] = 0;
                            }

                            queuePosition += 2;
                            ++queueLength;
```

```
                            break;
                        }

                    case '*':
                        {
                            bufQueue[queueLength] = (bufQueue[queuePosition] *
bufQueue[queuePosition + 1]) & 65535;
                            queuePosition += 2;
                            ++queueLength;
                            break;
                        }

                    case ':':
                        {
                            break;
                        }

                    case '>':
                        {
                            registers[inputQueue[i][1] - 97] =
bufQueue[queuePosition];
                            ++queuePosition;
                            break;
                        }

                    case '<':
                        {
                            bufQueue[queueLength] = registers[inputQueue[i][1] -
97];
                            ++queueLength;
                            break;
                        }

                    case 'J':
                        {
                            i = labels[inputQueue[i].Substring(1)];
                            break;
                        }

                    case 'C':
                        {
                            if (inputQueue[i].Length > 1)
                            {
                                char symbol = (char)(registers[inputQueue[i][1] -
97] % 256);
```

```
                                if (symbol == 10)
                                {
                                    output.WriteLine();
                                }
                                else
                                {
                                    output.Write(symbol);
                                }
                            }
                            else
                            {
                                char symbol = (char)(bufQueue[queuePosition] %
256);
                                if (symbol == 10)
                                {
                                    output.WriteLine();
                                }
                                else
                                {
                                    output.Write(symbol);
                                }
                                ++queuePosition;
                            }

                            break;
                        }

                    case 'Z':
                        {
                            if (registers[inputQueue[i][1] - 97] == 0)
                            {
                                i = labels[inputQueue[i].Substring(2)];
                            }

                            break;
                        }

                    case 'E':
                        {
                            if (registers[inputQueue[i][1] - 97] ==
registers[inputQueue[i][2] - 97])
                            {
                                i = labels[inputQueue[i].Substring(3)];
                            }
```

```
                            break;
                        }

                    case 'P':
                        {
                            if (inputQueue[i].Length > 1)
                            {
                                output.WriteLine(registers[inputQueue[i][1] -
97]);
                            }
                            else
                            {
                                output.WriteLine(bufQueue[queuePosition]);
                                ++queuePosition;
                            }

                            break;
                        }

                    case 'G':
                        {
                            if (registers[inputQueue[i][1] - 97] >
registers[inputQueue[i][2] - 97])
                            {
                                i = labels[inputQueue[i].Substring(3)];
                            }

                            break;
                        }

                    case 'Q':
                        {
                            isEnd = true;
                            break;
                        }

                    default:
                        {
                            bufQueue[queueLength] = int.Parse(inputQueue[i]);
                            queueLength++;
                            break;
                        }
                }
            }
```

```
                output.Close();
            }
        }
    }
```

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.109	24358912	1349803	250850
1	OK	0.031	10260480	69	6
2	OK	0.015	10309632	232	232
3	OK	0.062	10276864	3	0
4	OK	0.031	10215424	100	19
5	OK	0.031	11259904	56	58890
6	OK	0.031	10747904	67	30000
7	OK	0.015	10723328	67	30000
8	OK	0.031	10727424	55	30000
9	OK	0.031	10227712	461	62
10	OK	0.031	10469376	11235	21
11	OK	0.031	10657792	23748	42
12	OK	0.031	11419648	66906	9136
13	OK	0.031	10297344	7332	993
14	OK	0.031	10289152	4611	632
15	OK	0.031	10911744	37968	7332
16	OK	0.015	10231808	14	3
17	OK	0.031	10248192	70	14
18	OK	0.031	10227712	350	70
19	OK	0.031	10235904	1750	350
20	OK	0.031	10366976	8750	1750
21	OK	0.015	10952704	43750	8750
22	OK	0.062	14356480	218750	43750
23	OK	0.031	10825728	34606	4867
24	OK	0.046	18046976	683180	7