##### Задание 3

1. Написать набор функций для работы со стеком и очередью
2. Решить задания из модуля "Стеки и очереди".

Варианты заданий (номер варианта для подзадачи вычисляется как остаток от деления на 9 плюс 1 от варианта из [файла Варианты для ЛР №№2, 5, 6](https://edu.susu.ru/mod/resource/view.php?id=6115933) )

1. стек символов, очередь символов
2. стек целых чисел, очередь символов
3. стек вещественных чисел, очередь символов
4. стек символов, очередь целых чисел
5. стек целых чисел, очередь целых чисел
6. стек вещественных чисел, очередь целых чисел
7. стек символов, очередь вещественных чисел
8. стек целых чисел, очередь вещественных чисел
9. стек вещественных чисел, очередь вещественных чисел

typedef struct stack {

int top; // индекс верхнего элемента

тип\_по\_варианту data[100];

} stack;

typedef struct queue {

int first, size; // first - индекс первого значения в очереди, size - текущий размер очереди

// (first+size)%100 - индекс первого свободного места

тип\_по\_варианту data[100];

} queue;

Функции для работы со стеком

stack\_init(стек) // инициализация стека

stack\_push(стек, значение) // добавить значение в стек

stack\_pop(стек) // убрать верхнее значение

stack\_top(стек) // получить верхнее значение на стеке

stack\_isempty(стек) // проверить, что стек пуст

[Визуализатор стека](https://ipc.susu.ru/viz/stack.html)

Функции для работы с очередью

queue\_init(очередь) // инициализация очереди

queue\_add(очередь, значение) // добавить значение в очередь

queue\_drop(очередь) // убрать первое значение

queue\_first(очередь) // получить первое значение из очереди

queue\_isempty(очередь) // проверить, что очередь пуста

[Визуализатор очереди](https://ipc.susu.ru/viz/queue.html)

Пример main для целых чисел

int main()

{ stack s1;

stack\_init(&s1);

stack\_push(&s1,10);

stack\_push(&s1,20);

printf("%d\n",stack\_top(&s1)); // 20

stack\_pop(&s1);

if(!stack\_isempty(&s1))

printf("%d\n",stack\_top(&s1)); // 10

queue q1;

queue\_init(&q1);

queue\_add(&q1,10);

queue\_add(&q1,20);

printf("%d\n",queue\_first(&q1)); // 10

queue\_drop(&q1);

if(!queue\_isempty(&q1))

printf("%d\n",queue\_first(&q1)); // 20

}

**Критерии оценки**

* написаны функции для работы со стеком - 5 баллов
* написаны функции для работы с очередью - 5 баллов
* решено не менее 5 задач из модуля - 5 баллов (1 балл за задачу)

Максимальная оценка 15 баллов

Модуль "Стеки и очереди".

1. На доске имеется три колышка A, B и C. На колышек A нанизаны *N* одинаковых дисков, пронумерованных сверху вниз от 1 до *N*

. Колышки B и C пустые. Необходимо перенести все диски на колышек B, используя в качестве вспомогательного колышек C и соблюдая следующие правила:

* нельзя возвращать диски на колышек A;
* нельзя снимать диски с колышка B;
* можно перемещать только по одному диску за ход.

|  |  |
| --- | --- |
| начальное положение | конечное положение |

Определить, возможно ли получить заданный конечный порядок размещения дисков на колышке B и минимальное количество ходов для этого.

2. Шеф Виггам должен обеспечить правильный порядок движения платформ на городском параде. Платформы могут прибывать в любом порядке, но должны выходить на центральную площадь строго в порядке возрастания номеров. Виггам может направить платформу либо сразу на площадь, либо сначала на боковую улицу, а затем с нее на площадь. Длина боковой улицы достаточна для размещения всех платформ, но ширина улиц не позволяет одной платформе обгонять другую.

Напишите программу, определяющую, сможет ли Виггам обеспечить правильный порядок движения платформ на параде.

Первая строка ввода содержит одно целое число *N* (1 ≤ *N* ≤ 100) – количество платформ. Вторая строка содержит *N* различных целых чисел от 1 до *N* – номера платформ в порядке прибытия.

Вывести сообщение "YES", если можно обеспечить правильный порядок платформ, или сообщение "NO", если нельзя.

3. Горную вершину можно увидеть на фоне неба, если только она не скрывается за другой горой или ее контур не скрыт на фоне другой горы. Предположим, что склоны всех гор имеют наклон в 45°, и известны координаты и высоты вершин всех гор. Требуется определить количество видимых горных вершин.

Во входном файле в первой строке целое число *N*(1 ≤ *N* ≤ 10 000). Далее идет *N* строк, по два целых числа через пробел в каждой строке: координата *x**i* вершины (0 ≤ *x**i* ≤ 30 000) и ее высота *y**i* (1 ≤ *y**i* ≤ 10 000), последовательность вершин является упорядоченной по возрастанию координаты *x*: *x*1 ≤ *x*2 ≤ … ≤ *x**N*.

В выходной файл вывести количество видимых вершин.

4. Определить, является ли правильной последовательность из N круглых, квадратных и фигурных скобок, т.е. соответствует ли каждой открывающей скобке закрывающая и правильно ли они вложены.

5. Жители одного *N*-этажного дома решили устроить флэш-моб – изобразить ночью на стене дома ползущую "змейку" из *L*светящихся окон, включая и выключая свет в определённом порядке.

Они придумали схему движения змейки, которая представляет собой последовательность шагов R, L, U, D для движения вправо, влево, вверх и вниз соответственно. Если змейка достигает одного из краев стены, она выползает с другого края (если была наверху и ползла вверх – выползает снизу, и т.д.). Змейка должна выполнять один шаг в секунду.

Теперь нужно для каждого окна определить моменты включения и выключения света.

Окна пронумерованы слева направо снизу вверх, начиная с 1. На каждом этаже имеется по *M*

окон. В начальный момент времени видна только "голова" змейки, которая находится в первом окне. В течении первых *L*шагов в первом окне появляются последующие части змейки. Перед началом движения свет во всех окнах выключен, по окончании движения он также выключается.

Для каждого окна, в котором хотя бы раз должен зажечься свет, следует вывести его номер и интервалы времени, в течении которых свет должен гореть. Интервалы должны быть отсортированы по возрастанию времени начала. Если следующий интервал начинается сразу по окончании предыдущего, их следует объединить.