Задача 1. Реализирайте структура от данни стек. Реализирана от вас структура от данни трябва да осигурява следните основни методи и функции за Стек:

- *push*(*Object* o) – добавяне на нов елемент към върха на стека;
- *pop*() – изтриване (премахване, изваждане) и връщане на последния добавен елемент;
- *top*() – връща последния добавен елемент без да го изтрива;
- *size*() – връща броя на елементите в стека;
- *isEmpty*() – връща булева стойност, показваща дали има елементи в стека (или е празен).

Да се състави алгоритъм и програма на Java (словесно описание на алгоритъма и съответстващия програмен текст), която при въвеждане на два стека от числа, подредени в ***намаляващ ред от върха към дъното***, използвайки операциите *push*() и *pop*() построява нов стек, който се състои от всички елементи на дадените два стека, подредени в ***нарастващ ред от върха към дъното***.

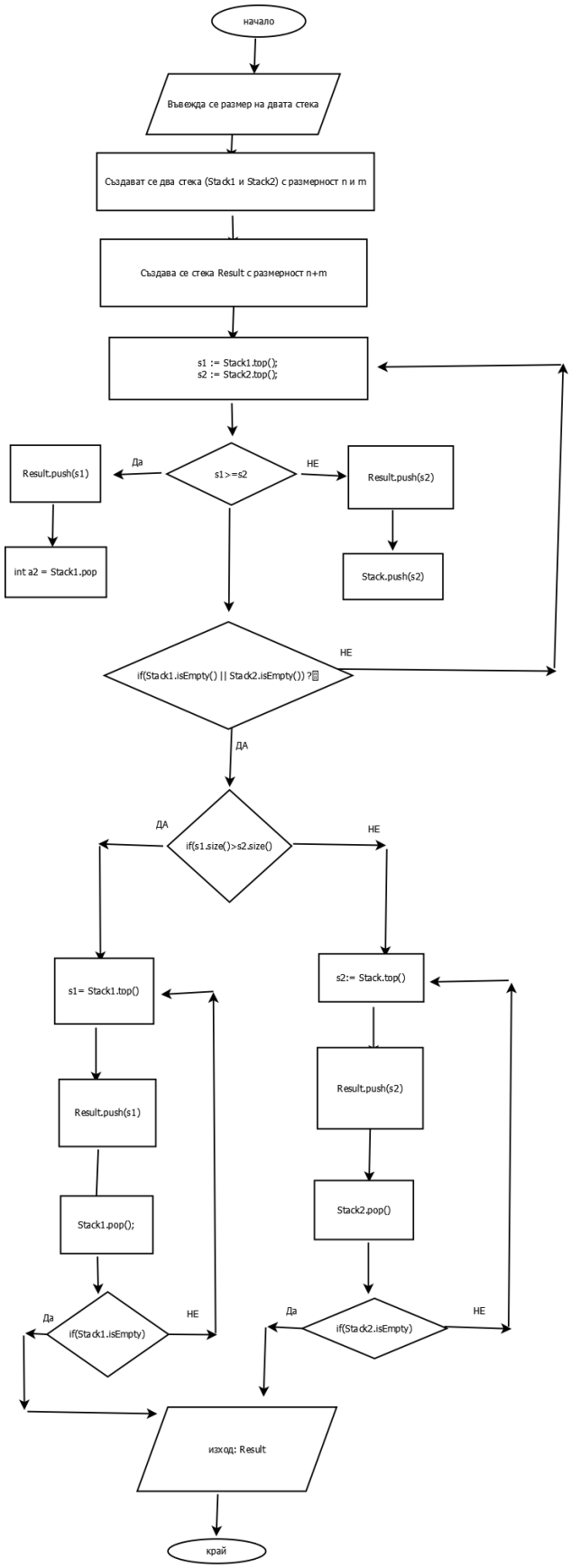
Вход: Стандартният вход съдържа три реда. На първият ред се задават две цели числа *n* и *k* разделени с интервал. На вторият ред се въвежда редица от *n*, които че се съхраняват във първият стек, а на третият ред се задава редица от *k*, които ще се съхраняват във вторият стек. За разделител между числата използвайте интервал.

Изход: На единствен ред на стандартния изход изведете в квадратни скоби [] елементите на полученият (трети) стек, сортирани във възходящ ред.

Примерен Вход	**Примерен Изход**
5 6 8, 6, 5, 3, 1 9, 7, 6, 3, 2, 1	[1, 1, 2, 3, 3, 5, 6, 6, 7, 8, 9]

Алгоритъм за решаване на задачата:

1. Въвежда се размерност на двата стека (стойности за *n* и *m*);
2. Създават се два стека (*Stack1* и *Stack2*) с размерности *n* и *m*;
3. Създава се стека *Result* с размерности *n* + *m*;
4. Във низходящ ред се въвеждат и се съхраняват *n* на брой числа в *Stack1*;
5. Във низходящ ред се въвеждат и се съхраняват *m* на брой числа в *Stack2*;
6. *s1* : = *Stack1.top*();
7. *s2* := *Stack2.top*();
8. Ако *s1* ≥ *s2, Result.push*(*s1*) и *Stack1.Pop*(). В противен случай *Result.push*(*s2*) и *Stack2.pop*().
9. Ако *Stack1.empty()* или *Stack2.empty*(), Стъпка 10. В противен случай Стъпка 6.
10. Ако *Stack1.size()* > *Stack2.size()*:
    - 10.1. *s1* : = *Stack1.top*();
    - 10.2. *Result.push*(*s1*);
    - 10.3. *Stack1.pop*()
    - 10.4. Ако *Stack1.empty(),* Стъпка 11. В противен случай Стъпка 10.1.
11. Ако *Stack2.size()* > *Stack1.size()*:
    - 11.1. *S2* : = *Stack2.top*();
    - 11.2. *Result.push*(*s2*);
    - 11.3. *Stack2.pop*()
    - 11.4. Ако *Stack2.empty(),* Стъпка 12. В противен случай Стъпка 11.1.
12. Извежда се Result.


начало
Въвежда се размер на двата стека
Създават се два стека (Stack1 и Stack2) с размерност n и m
Създава се стека Result с размерност n+m
s1 := Stack1.top();
s2 := Stack2.top();
s1>=s2
Да
НЕ
Result.push(s1)
int a2 = Stack1.pop
Result.push(s2)
Stack.push(s2)
if(Stack1.isEmpty() || Stack2.isEmpty()) ?
НЕ
ДА
if(s1.size()>s2.size()
ДА
НЕ
s1= Stack1.top()
Result.push(s1)
Stack1.pop();
if(Stack1.isEmpty)
Да
НЕ
s2:= Stack.top()
Result.push(s2)
Stack2.pop()
if(Stack2.isEmpty)
Да
НЕ
изход: Result
край