PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA

ALGORITMIA
Laboratorio 3
2014-2

Indicaciones generales:

- Duración: 2h 45 min.
- Puntaje máximo debido a esta prueba: 10
- Puntaje máximo debido a trabajo grupal: 10
- No se permite el uso de material de consulta.
- No debe resolver todas las preguntas del cuestionario, debe resolver las preguntas de manera que su puntaje sume 10 puntos o más.
- Si su puntaje total suma más de 10, su nota en esta prueba será de 10.
- Cada vez que desee que una pregunta sea calificada deberá llamar a su JP asignado.
- Puede llamar al JP para calificar alguna pregunta sólo hasta las 10:45 am, no se realizará revisiones pasada la hora indicada.
- Se calificará al momento que la solución pase correctamente todos los casos de prueba brindados por el JP.
- Si la solución no pasa algún caso de prueba, no se otorga ningún puntaje por la pregunta en ese momento; sin embargo, en una revisión posterior podría asignarse puntaje si la nueva solución pasa todos los casos de prueba.

Question 1 (1 point) Pancake Sort

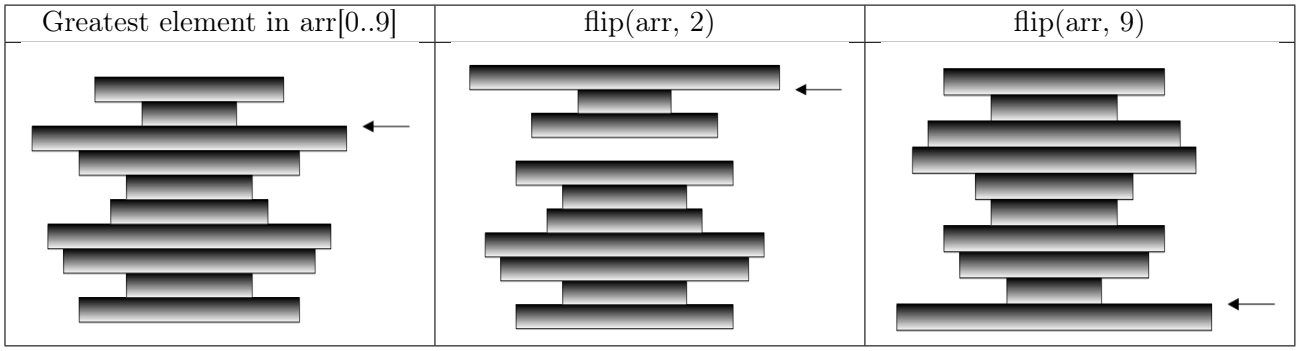
Instructions

Pancake sort refers to the process of sorting by size a disordered stack of pancakes when a spatula can be inserted at any point in the stack to flip all pancakes above it. A **flip** is a basic operation and it can be **defined by an index**, e.g. flip(arr, 2) means "flip all the elements in arr from 0 to 2". Implement pancake sort using this algorithm:

For N from the size of the array to 2:

- Find the index of the greatest element in arr[0..N-1], let us call it "pivot".
- Flip the array from the first element to the "pivot".
- Flip the array from the first element to N-1.

For each iteration the greatest element is pulled to the last position, that is why the size of the array that is being sorted must be reduced each iteration. The output must be the **sequence of flips** performed to sort the stack of pancakes. Remember that **each flip is defined by an index** and note that this algorithm might perform unnecessary flips.



For each test case the first digit represents the number of elements to be sorted.

Sample Input

```
3  50  100  25
10  3  8  10  1  2  9  4  6  5  7
5  6  7  8  9  10
```

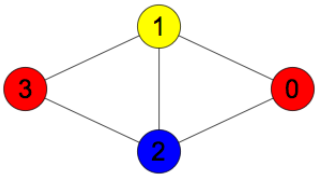
Sample Output

```
1  2  1  1
2  9  4  8  0  7  3  6  4  5  0  4  0  3  0  2  0  1
4  4  3  3  2  2  1  1
```

Question 2 (5 points) M Coloring

Instructions

Given an undirected graph and an integer m, determine a color configuration such that, using at most m colors, no two adjacent vertices are colored with the same color. For example, for m = 3, the following graph can be colored successfully (each node has a number that indicates that it is the i-th node of the graph):



Assuming that each color is represented by a number (Red = 0, Yellow = 1, Blue = 2), the color configuration for this graph is 0 1 2 0 (the i-th element is the color assigned to the i-th node).

The process must assign one color to each node, starting with node 0, then node 1, then node 2, etc. Each assignment must try to use the color represented by the lowest possible number.

The first line of the input will contain m (the maximum number of colors that can be used). The second line will contain the number of nodes in the graph and the following lines contain the connections in the graph as pairs i j that represent a connection between node i and j. The line -1 -1 means that all connections of the graph have been read. The output must be a color configuration as given for the previous graph. If no color configuration exists, the output must be X.

Sample Input

```
3
4
3 2
0 1
1 2
0 2
1 3
-1 -1

2
3
0 1
1 2
2 0
-1 -1
```

Sample Output

```
0 1 2 0
X
```

Question 3 (1 point) Bubble Sort

Instructions

Print the partially sorted arrays that result in each iteration of bubble sort.
For each test case the first digit represents the number of elements to be sorted.

Sample Input

```
9  12  3  100  6  43  18  8  12  2
4  200  60  20  8
4  −200  −60  −20  −8
```

Sample Output

```
3  12  6  43  18  8  12  2  100
3  6  12  18  8  12  2  43  100
3  6  12  8  12  2  18  43  100
3  6  8  12  2  12  18  43  100
3  6  8  2  12  12  18  43  100
3  6  2  8  12  12  18  43  100
3  2  6  8  12  12  18  43  100
2  3  6  8  12  12  18  43  100
2  3  6  8  12  12  18  43  100

60  20  8  200
20  8  60  200
8  20  60  200
8  20  60  200

−200  −60  −20  −8
−200  −60  −20  −8
−200  −60  −20  −8
−200  −60  −20  −8
```

Question 4 (3 points) Sierpinski Carpet

Instructions

The construction of the Sierpinski carpet can be defined recursively:

- Start with a square.
- Cut the square into 9 congruent subsquares and remove the central subsquare.
- Apply the previous step recursively to the remaining 8 subsquares.

Implement a recursive program that, for any given number n, prints a matrix of characters that represents the Sierpinski carpet after applying the third step n times.

Sample Input

```
0
1
2
```

Sample Output

```
XXX
X X
XXX

XXXXXXXXX
X XX XX X
XXXXXXXXX
XXX   XXX
X X   X X
XXX   XXX
XXXXXXXXX
X XX XX X
XXXXXXXXX

XXXXXXXXXXXXXXXXXXXXXXXXXXX
X XX XX XX XX XX XX XX XX X
XXXXXXXXXXXXXXXXXXXXXXXXXXX
XXX   XXXXXX   XXXXXX   XXX
X X   X XX X   X XX X   X X
XXX   XXXXXX   XXXXXX   XXX
XXXXXXXXXXXXXXXXXXXXXXXXXXX
X XX XX XX XX XX XX XX XX X
XXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXX         XXXXXXXXX
X XX XX X         X XX XX X
XXXXXXXXX         XXXXXXXXX
XXX   XXX         XXX   XXX
X X   X X         X X   X X
XXX   XXX         XXX   XXX
XXXXXXXXX         XXXXXXXXX
X XX XX X         X XX XX X
XXXXXXXXX         XXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXX
X XX XX XX XX XX XX XX XX X
XXXXXXXXXXXXXXXXXXXXXXXXXXX
XXX   XXXXXX   XXXXXX   XXX
X X   X XX X   X XX X   X X
XXX   XXXXXX   XXXXXX   XXX
XXXXXXXXXXXXXXXXXXXXXXXXXXX
X XX XX XX XX XX XX XX XX X
XXXXXXXXXXXXXXXXXXXXXXXXXXX
```

Question 5 (1 point) Happy Number

Instructions

Starting with an integer, replace it by the sum of the squares of its digits and repeat the process until the number equals 1 or it loops endlessly in a cycle which does not include 1. If the process ended with 1, the original integer is said to be happy; otherwise, it is said to be unhappy.

It is known that the process described above (constantly replacing an integer by the sum of the squares of its digits) always eventually reaches one of these 10 numbers: 0, 1, 4, 16, 20, 37, 42, 58, 89, or 145. Implement a recursive program that returns 1 if a given number is happy and 0 if it is unhappy.

Sample Input

```
3
10
11
31
17
14
```

Sample Output

```
0
1
0
1
0
0
```

Question 6 (5 points) Sudoku

Instructions

Implement a backtracking algorithm that, for a given partially filled 9×9 matrix, assigns digits (from 1 to 9) to the empty cells so that every row, column, and sub grid of size 3×3 contains exactly one instance of the digits from 1 to 9.

Sample Input

```
3 0 6 5 0 8 4 0 0
5 2 0 0 0 0 0 0 0
0 8 7 0 0 0 0 3 1
0 0 3 0 1 0 0 8 0
9 0 0 8 6 3 0 0 5
0 5 0 0 9 0 6 0 0
1 3 0 0 0 0 2 5 0
0 0 0 0 0 0 0 7 4
0 0 5 2 0 6 3 0 0

5 0 0 0 8 0 0 4 0
0 9 0 0 0 5 7 1 0
4 0 7 1 0 0 0 0 0
0 0 0 0 3 0 0 0 4
0 0 0 0 6 0 0 7 0
9 0 8 0 0 0 3 0 6
0 0 9 0 0 8 0 0 0
0 4 0 0 7 0 5 0 0
0 0 3 4 1 6 9 8 0
```

Sample Output

```
3 1 6 5 7 8 4 9 2
5 2 9 1 3 4 7 6 8
4 8 7 6 2 9 5 3 1
2 6 3 4 1 5 9 8 7
9 7 4 8 6 3 1 2 5
8 5 1 7 9 2 6 4 3
1 3 8 9 4 7 2 5 6
6 9 2 3 5 1 8 7 4
7 4 5 2 8 6 3 1 9

5 3 1 6 8 7 2 4 9
2 9 6 3 4 5 7 1 8
4 8 7 1 9 2 6 5 3
6 7 5 2 3 1 8 9 4
3 2 4 8 6 9 1 7 5
9 1 8 7 5 4 3 2 6
1 6 9 5 2 8 4 3 7
8 4 2 9 7 3 5 6 1
7 5 3 4 1 6 9 8 2
```

Question 7 (3 points) Word Wrap

Instructions

Word wrap is the feature of most text editors, word processors, and web browsers that breaks lines between and not within words, which generates additional spaces. The sum of squares of the number of spaces is used to measure how much space is wasted. Use dynamic programming to minimize this.

e.g.
Assume we want to wrap the string *aaa bb cc ddddd* to a line width of 6 characters.

Line width: 6 - Greedy Solution		Dynamic Programming Solution	
aaa bb	Remaining space: 0	aaa	Remaining space: 3
cc	Remaining space: 4	bb cc	Remaining space: 1
ddddd	Remaining space: 1	ddddd	Remaining space: 1
cost $= 0^2 + 4^2 + 1^2 = 17$		cost $= 3^2 + 1^2 + 1^2 = 11$	

The first line of the input is n m, where n represents the number of words and m represents the line width. The next n lines: l w, where l (integer) represents the length of w (string).

Sample Input

```
6 12
2 We
4 love
8 computer
7 science
3 and
11 mathematics

6 10
8 Everyone
4 must
5 learn
3 how
2 to
4 code
```

Sample Output

```
We love
computer
science and
mathematics

Everyone
must learn
how to code
```

Question 8 (1 point) H-Sequence

Instructions

An *h-sequence* is made of:

- one 0, or
- one 1 followed by two *h-sequences*

An *h-sequence* can be defined by the following grammar:

$$\langle h \rangle = 0|1\langle h \rangle\langle h \rangle$$

Implement a recursive program that returns 1 if a given array is an *h-sequence* and 0 otherwise. For each test case the first digit represents the number of elements to be sorted.

Sample Input

```
1 0
3 1 0 0
1 1
5 1 0 1 0 0
5 1 1 0 1 0
9 1 1 0 0 1 0 1 0 0
7 1 0 1 0 1 0 1
```

Sample Output

```
1
1
0
1
0
1
0
```