TECNOLÓGICO SUPERIOR DE JALISCO
MARIO MOLINA
ZAPOPAN

Data structures

Linear Data Structures: Stacks

Deadline: 11/10/2017.

Objective of the practice

To use a stack to do numerical coding in different bases.

Background

A *stack* is a *linear data structure* of the LIFO type (*Last–In, First–Out*) in which the data is added and removed by just one side. You must use a stack for integer numbers to solve the *number coding problem* in base r, $2 \leq r \leq 16$.

Number codification

In a *positional coding system* (PCS) a number is encoded as a sequence of symbols called *digits*. The sequence of digits is called a *numeral*. The value represented by each digit depends not only on itself but also on its position relative to other digits. A PCS is based on a positive integer *radix* (or *base*) $r \geq 2$ and the digit set $\{0,1,\cdots,r-1\}$. An unsigned integer is represented by a vector of length k where:

$$x_k x_{k-1} \cdots x_2 x_1 = \sum_{i=1}^{k} x_i \cdot r^{i-1}$$

The term $x_i \cdot r^{i-1}$ is the corresponding to the i–th position where:

- x_i is the i–th digit (from right to left).
- r^{i-1} is the *positional value*.
- $x_i \cdot r^{i-1}$ is the *actual value* of the i–th digit.

For example, let $n = 100111011$ be a *binary* numeral (i.e., with $r = 2$; we can also write 100111011_2 to denote it). Then:

$$\begin{aligned} n &= 1 \cdot 2^8 + 0 \cdot 2^7 + 0 \cdot 2^6 + 1 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 \\ &= 256 + 32 + 16 + 8 + 2 + 1 \\ &= 315 \end{aligned}$$

Now we say that $100111011_2 = 315$. We call the leftmost digit (x_k) the *most significative digit* (MSD) due to its position corresponds to the greater positional value.

Number coding in radix *r*

Assume an integer $n \geq 0$. To encode n in radix r you must implement the *algorithm of remainders*, which can be stated as follows:

1. Calculate $x \leftarrow n \bmod r$ ($n \bmod r$ is the *remainder* of the euclidean division (*integer division*) of n by r, also called *the modulo r of n*). Reserve x as the MSD of your numeric encoding.
2. Calculate $n \leftarrow n \div r$ ($n \div r$ is the integer division of n by r).
3. Repeat from step 1 while $n > 0$.
4. The algorithm ends and provides as a result the sequence of reminders that was generated.

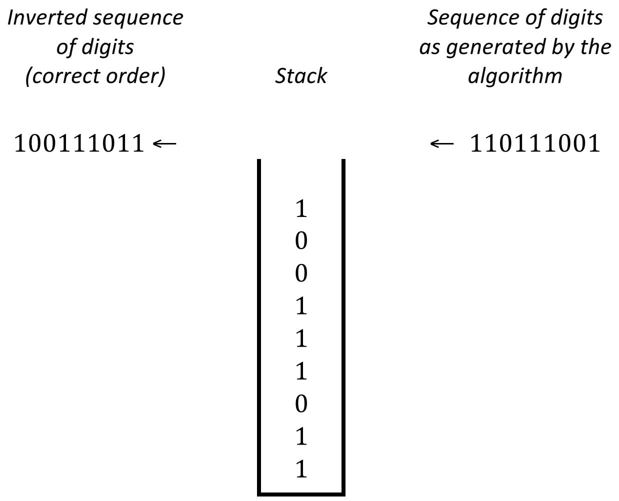
Por example, suppose you want to code 315 to the radix 2 ($r = 2$). Then you have the following run:

	n	x	
Beginning →	315	1	
	157	1	
	78	0	
	39	1	
	19	1	
	9	1	
	4	0	
	2	0	
	1	1	← The last digit is the MSD
End →	0		

Therefore, $315 = 100111011_2$.

Implementation

You must program the algorithm of remainders to code a nonnegative integer to any radix $2 \leq r \leq 16$. Note that the encoding algorithm generates an inverted sequence. You must reverse the sequence using stack. For example, for $n = 315$ we have the following:



You must implement your stack *dinamically*,
i.e., on a linked list. Do not limit its capacity.

Let x be a digit. Then $0 \leq x < r$. E.g., if $r = 2$ then $0 \leq x < 2$ (i.e., $x = 0$ or $x = 1$) and if $r = 10$, then $0 \leq x < 10$. For a radix greather than 10 ($11 \leq r \leq 16$), we use the following aditional symbols:

x	Symbol	x	Symbol
10	A	13	D
11	B	14	E
12	C	15	F

E.g., for the hexadecimal encoding of 43006 ($r = 16$), the algorithm of remainders generates the succession 14, 15, 7 and 10. According to the suggested symbology, this sequence is E, F, 7 and A. Then, the hexadecimal coding of 43006 is $A7FE$.

Operation of your program: Your program must ask for a nonnegative number n and a radix r in the range of 2 to 16. In response it must show on screen the codification of n with radix r.

Examples

1. A run of your program should look like:

```
Number:   2017
Radix:    7

Encoding: 5611
```

2. Assume $n = 7791$. Below is the coding to each of the possible radices:

r	Encoding	r	Encoding
		9	11616
2	1111001101111	10	7791
3	101200120	11	5943
4	1321233	12	4613
5	222131	13	3714
6	100023	14	$2BA7$
7	31500	15	2496
8	17157	16	$1E6F$

Delivery

Send your code in a .zip, .rar or .7z file named `encoding.`*<name>*`.zip`, where *<name>* is *your name*. For example: `encoding.tyron.lannister.zip`. Send your file to aaguilar.itszapopan@gmail.com no later than the deadline with the subject *Encoding*. They will not be reviewed consignments that fail to complay with any of these requirements or if they do not compile. The deadline is on Friday, November 10th.