```
$Id: lab3c-scanf-arrays.mm,v 1.8 2012-01-24 20:58:22-08 - - $
/afs/cats.ucsc.edu/courses/cmps012b-wm/Labs-cmps012m/lab3c-scanf-arrays
```

In this lab, you will wrote a program in both Java and in ANSI C that will interpret input as Reverse Polish Notation expressions.

1. Java version

Write a program in Java called **`jrpn.java`** that will do the following:

(a) Create a stack represented as an array:
```
final int EMPTY = -1;
int top = EMPTY;
double stack = new double[16];
```
Of course, you will never use the number 16 anywhere else in the program, instead using **`stack.length`**

(b) Loop over the standard input reading words one at a time using **`hasNext()`** and **`next()`**. Stop at end of file.

(c) If this can be converted into a **`double`** by means of **`Double.parseDouble`**, push the number onto the stack. If this causes tack overflow, print an error message and discard the number.

(d) Otherwise, the word is a single character operator. Print an error message and discard it if it is not a single character. If it is, extract the first charcter from the string using **`charAt(0)`**.

(e) Use a **`switch`** statement to perform one of the following operations:

(i) If it is a **`'+'`**, **`'-'`**, **`'*'`**, or **`'/'`**, pop two numbers off the stack, the right operand first, and the left operand next, and push the value of the result back on the stack. Print an error messwage and do nothing if there is tack underflow.

(ii) If the operator is a semi-colon (**`';'`**), print the stack elements, one per line, starting from bottom to top (upside down), using the format **`"%22.15g"`**.

(iii) Otherwise print an error message.

(f) Note that you do not need to do any checking concerning numbers outside the range of IEEE-754 floating point arithmetic, nor should you check for division by zero. When your program is working, try it and see.

2. C version

Now translate your program into a C program called **`crpn.c`**. Your program should follow roughly the same sequence of steps. Some notes:

(a) Study the example programs in this directory.

(b) You can create the stack variables using the declarations
```
#define STACK_SIZE 16
#define EMPTY (-1)
double stack[STACK_SIZE];
int top = EMPTY;
```

(c) Study the program **`scanf.c`** in this directory. It reads words from **`stdin`**, prints the words and their lengths, and then decides if the input contains a number.

(d) Use **`strlen(3)`** to determine the length of the string that was read, and you can use the following statement to extract its first character:
```
char operator = buffer[0];
```

3. What to submit

Submit **`jrpn.java`** and **`crpn.c`**, and the required pair programming files, if appropriate.