roll_dir

A little project of demos / exercises writing the same simple program in a variety of different languages. For this, I've taken some inspiration from TTRPGs like *Pathfinder* and chosen to write a program called `roll` that simulates a specified number of rolls for a regular-polyhedral die with a specified number of sides and then adds an optionally-specified bonus. Syntax varies by language; but the general idea is to call either `roll N` for a single roll, or `roll K N B` for multiple rolls with optional bonus.

For example, in the lingo:

- To roll 1d20, simply call `roll 20`.
- To roll 3d6 with a bonus of 4, call `roll 3 6 4`.

Setup & usage instructions for each one are described below. Required arguments are indicated by angled brackets `<>`, and optional arguments are indicated by square brackets `[]`. Output in most cases is printed according to the folowing format:

```
KdN + {B}  ==>  roll1 + roll2 + ... + {B}  =  TOTAL
```

NOTE: The instructions here generally assume you are running these commands in the same directory as the indicated *roll* file(s). If not, simply add the path in front of the filename as required. For example, if the files are located in `~/useful_bash_stuff/bin/roll_dir` and you are at a shell prompt in your home directory, you would call `roll.sh` by typing `useful_bash_stuff/bin/roll_dir/roll.sh`, etc.

Enjoy!

~ *duvall3*

Shell Script ~ *roll.sh*

Usage, at shell prompt:

```
roll.sh <N>
```

```
roll.sh <K> <N> [B]
```

Setup: Simply place the file in a directory belonging to your `$PATH` (or, alternately, add the relevent directory to your `$PATH` — my personal favorite method for this is `export PATH=$(pwd):$PATH`); and if necessary, run `chmod +x roll.sh` .

C++ ~ *roll.cpp, roll*

Usage, at shell prompt:

```
roll <N>
```

```
roll <K> <N> [B]
```

Setup: The program should be ready to compile and should not require any special options. For example, using *G++*:

```
g++ -o roll roll.cpp
```

Then follow the same instructions for `chmod` and `PATH` as in `roll.sh` above.

ROOT ~ *roll.cxx*

Usage, at *ROOT/CINT* prompt:

```
roll(<N>);
```

```
roll(<K>, <N>, [B]);
```

Setup: Place the file (or a link to the file) in a directory where your *ROOT* installation can find it — `$ROOTSYS/macros` is usually a safe bet. Then at the *CINT* prompt, simply load the macro with the command `.L roll.cxx`, then call the function as indicated above. Alternatively, you can add the following line to your *ROOT* logon script (sometimes called `rootlogon.C`), and it will load automatically for you whenever you start *ROOT*:

```
gROOT->LoadMacro("roll.cxx");
```

Python ~ *roll.py*

Usage, at *Python* prompt:

```
roll(<N>)
```

```
roll(<K>, <N>, [B])
```

Setup: Place `roll.py` in a directory where your *Python* installation can find it (try `echo $PYTHONPATH` at the shell prompt if you need some places to look). Then start *Python* and either import the "roll" module or just the "roll" function. To make usage match the above, make your import line as follows:

```
from roll import roll as roll
```

Octave / MATLAB

Usage, at *Octave* prompt:

```
roll(<N>);
```

```
roll(<K>, <N>, [B]);
```

Setup: Simply place the file somewhere that *Octave* can find. Within *Octave*, you can either: 1) check the current path by running the `path` function; or 2) add the directory containing `roll.m` to *Octave*'s path using the `addpath` function. (See `doc path` and `doc addpath` for more information).

VIM ~ *roll.vim, roll_script.vim*

As Script ~ *roll_script.vim*

Usage, at shell prompt:

`roll_script.vim <N>`

`roll_script.vim <K> <N> [B]`

Setup: Same as for `roll.sh` above.

As VIM Function ~ *roll.vim*

Usage, in *VIM* Normal mode:

`:echo Roll(<N>)`

`:echo Roll(<K>, <N>, [B])`

Setup: In *VIM* Normal mode, source the file using `:source roll.vim` . Then, call the function within *VIM* as you would any other *VIM* function, as indicated above.

AWK ~ *roll.awk, roll.txt*

Operating on Keyboard Input (*stdin*)

This will effectively start *AWK* in a sort of "roll-command" mode, where each line you type will interpreted as an argument or set of arguments to `roll`.

Usage, from shell prompt:

`awk -f roll.awk`, then

`<N>` or `<K> <N> [B]`

Setup: None!

Operating on Input File (*roll_args.txt*)

This will instruct *AWK* to read a set of `roll` inputs from a text file. *AWK* will read each line of the input file as a set of arguments to the `roll` function. This could be used to quickly run (and re-run) a given set of rolls. An example file, `roll_args.txt`, is provided. The example file will roll 1d20, 2d6, and 3d4 with a bonus of 5.

Usage, from shell prompt:

`awk -f roll.awk roll_args.txt`

Setup: None!

Happy Hacking! ~ *duvall3*