CSC254: Project 1
Comparing Languages

In this project, you will solve a simple programming problem in each of five different programming languages (six for CSC454 students).

Pascal's Triangle

A *binomial coefficient*, written $\binom{n}{k}$ for integers $0 \leq k \leq n$, is the coefficient of the x^k term in the polynomial expansion of $(1+x)^n$. Binomial coefficients occur throughout mathematics and computing. For example, you should know that $\binom{n}{k}$ is the number of subsets of size k from a set of n elements ("n choose k").

Work out the expansion of $(1+x)^n$ for the first few values of n. Write the coefficients for each row on a different line. You should notice a pattern.

DO THAT NOW, THEN KEEP READING.

Pascal's Triangle is a triangular table where row n contains the binomial coefficients for that n and all $0 \leq k \leq n$. It also illustrates the relationship between the coefficients. This is easiest to see if you write the rows centered rather than left-aligned.

DO THAT NOW, THEN KEEP READING.

The first row contains just a single entry, $\binom{0}{0}$, which is the coefficient of the x^0 term in the expansion of $(1+x)^0$. Since $(1+x)^0 = 1$ and the x^0 term is the constant term, $\binom{0}{0} = 1$.

MAKE SURE YOU UNDERSTAND THAT, THEN KEEP READING.

After the first row, each binomial coefficient $\binom{n}{k}$ is the sum of the numbers "above" it. That is, each binomial coefficient is the sum of the binomial coefficient above and to the left of it and above and to the right of it. The row above row n is row $n-1$. The position above and to the left is $k-1$ in that row and the position to the right is k.

With that observation, can you come up with a recursive definition of the binomial coefficient $\binom{n}{k}$?

TRY IT NOW BEFORE TURNING TO THE NEXT PAGE

DID YOU TRY IT YOURSELF BEFORE TURNING TO THIS PAGE? BECAUSE YOU WON’T LEARN ANYTHING IF YOU DON’T TRY TO DO IT YOURSELF.

Each binomial coefficient is the sum of the binomial coefficient above and to the left of it and above and two the right of it. The binomial coefficient above and the left is $\binom{n-1}{k-1}$ and the one above and to the left is $\binom{n-1}{k}$. So:

$$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$$

for any integer $n > 0$ and $0 \leq k \leq n$.

That is so cool.

There is also an iterative method for computing binomial coefficients, and there is a method that uses the factorial function (which itself has natural recursive and iterative versions and which you may have learned back in your discrete math class).

Project Requirements

You **MUST** write programs that print the first n rows of Pascal's Triangle using the following programming languages:

- Imperative: C or C++, Java, Rust
- Functional: Lisp, OCaml
- CSC454 students must also use Haskell

For programs in functional languages, your implementation **MUST** be based on the **recursive** definition of the binomial coefficients. For the other languages you can use whatever method you like (but recursion works fine in them also).

Your programs **MUST** get the value of n from the command-line. Note that it will be a string, which you **MUST** convert into an integer.

Your programs **MUST** print the triangle with each row left-aligned on a separate line and each coefficient separated by a single space.

You **MUST** submit a single ZIP archive containing a README file and the source code for all of your programs.

We will build and run your programs **on the CSC instructional machines** using the instructions in your README.

Programs that do not build will receive a grade of **0** (zero).

Programs that crash will receive a grade of **0** (zero).

Late submissions without an approved accommodation will receive a grade of **0** (zero).

Detailed requirements and additional information follows.

README

You **MUST** submit a README file in text or PDF format.

This document **MUST** include:

- Name and NetID for all members of the team (maximum two)
- Instructions for building **AND** running your programs from the command-line using the tools available on the CSC instructional machines

It is **your job** to make it clear and easy for us to cut-and-paste from your README to build and run your programs.

Java

The Java compiler is `javac` and to run Java programs you use the `java` program, as in:

```
java TheClassToRun foo bar baz
```

Here "`TheClassToRun`" is the name of the class whose `main` method will be run, and any additional arguments become the `String[]` argument to that method.

I hope that you knew that already.

C/C++

You may use either C or C++. You should all know C from your previous courses. I suggest learning some C++ if you don't know it already, but it is not required.

For C, the compiler on the CSC instructional machines is `gcc`. For C++, it is `g++`. You probably knew that already.

You **MUST** use the following compiler options: "`-Wall -Werror`".

If you do not do your development on the CSC instructional machines, I **STRONGLY** suggest that you test both building and running on the CSC instructional machines. No excuses.

For C and C++, the command-line arguments become the `argc` and `argv` arguments to `main`. Note that `argv[0]` is the name of the executable program. I hope that you knew that already.

Rust

The Rust compiler is `rustc`.

For Rust programs, the command-line arguments are in `env::args()` which is defined in the `std::env` module. As in C, the first element of that array is the name of the program being executed.

Lisp

For this course, we will use the latest version of Armed Bear Common Lisp (`abcl.org`).

ABCL runs on the Java Virtual Machine (JVM). To run it, use:

```
    java -jar path/to/abcl.jar
```

where you provide the path to the ABCL jar downloaded from its website.

For the help message, give "`--help`" after the jarfile. Check out the "`--load`", "`--eval`", and "`--batch`" options.

Any remaining arguments become the command-line arguments to the Lisp program. Note that if the first such argument starts with a hyphen ("`-`"), you must precede it with "`--`".

Those command-line arguments, if any, are available as the value of the variable `EXTENSIONS:*COMMAND-LINE-ARGUMENT-LIST*`, which will be a list.

The CSC instructional network proves CMU Common Lisp (CMU CL) via the `lisp` command. That should be fine (any Common Lisp implementation should be fine), but we will use ABCL and so should you.

OCaml

To run your code under the `ocaml` interpreter and pass the numbers 3, 1, and 2 as command-line arguments, use:

```
    ocaml project1.ml 3 1 2
```

You may also compile with `ocamlc` to produce native binaries.

In OCaml, the command-line arguments are the value of `Sys.argv`. This will be an array whose first element is the name of the program being run. You may find the OCaml library function `Array.to_list` helpful in working with it.

Haskell

The Haskell compiler is `ghc`. There is also an interpreter: `ghci`.

In Haskell, the `System.Environment` module provides access to the command-line arguments:

- Computation `getProgName` has type "`IO String`" and returns the name of the program being run
- Computation `getArgs` has type "`IO [String]`" and returns a list of the arguments given to the program on the command-line

Policies

Late Policy

Late projects will receive a grade of 0. You **MUST** submit what you have by the deadline. If there are extenuating circumstances, submit what you have before the deadline and then explain yourself via email.

If you have a medical excuse (see the course syllabus), submit what you have and explain yourself as soon as you are able.

Collaboration Policy

I assume that you are in this course to learn. You will learn the most if you do the projects **yourself**.

That said, collaboration on projects is permitted, subject to the following requirements:

- Teams of no more than **2** (**two**) students, all currently taking CSC254/454.
- You **MUST** be able to explain anything you or your team submit, IN PERSON AT ANY TIME, at the instructor's or TA's discretion.
- One member of the team should submit code on the team's behalf in addition to their writeup. The other team member **MUST** submit a README (only) indicating who their teammate is.
- Both members of a collaborative team will get the same grade on the project.

Working in a team only works if you actually do all parts of the project together. If you only do one half of the project, then you only learn one half of the material. If one member of a team doesn't do their part or does it incorrectly (or dishonestly), all members pay the price.

Academic Honesty

I assume that you are in this course to learn. You will learn nothing if you do not do the projects **yourself**.

Do not copy code from other students or from the Internet.

Avoid Github and StackOverflow completely for the duration of this course.

The use of generative AI tools is **NOT** permitted in this course.

Do **NOT** submit code that you can't explain.

Posting homework and project solutions to public repositories on sites like GitHub is a violation of the University's Academic Honesty Policy, Section V.B.2 "Giving Unauthorized Aid." Honestly, no prospective employer wants to see your coursework. Make a great project outside of class and share that instead to show off your chops.