# Assignment 2 – Slice of Pi

Jonathan Cheng

CSE 13S – Spring 2023

## Purpose

The purpose of this assignment is to gain practical experience in computing the Taylor series expansions for some function f(x) using command line options. Specifically, this assignment will implement a program that takes as input a function f(x) and its desired degree of approximation n, and outputs the corresponding Taylor series expansion up to degree n.

By completing this assignment, us, the students will learn how to:

1. Compute Taylor series expansions for functions using finite differences and factorials
2. Implement command line options to enable user input and program customization

## How to Use the Program

Usage: ./program [OPTIONS] FILENAME

Compute some function for a given input using command line options.

OPTIONS -a Runs all tests. -e Runs e test. -b Runs BBP pi test. -m Runs Madhava pi test. -r Runs Euler pi test. -v Runs Viete pi test. -w Runs Wallis pi test. -n Runs Newton square root tests. -s Print verbose statistics. -h Display program synopsis and usage.

Arguments:
FILENAME aebmrvwnsh

Make Targets: all Build the program with default options. clean Remove all intermediate files and the program binary. help Display this help message and exit.

```
Examples:
    $ make
    $ ./program function.txt
    $ ./program -d 10 -x 2.5 -o result.txt function.txt
    $ ./program -v function.txt
```

# Program Design

Different sections of the program in which each play a significant role.

## 0.1 Getopt Section

1. Parse command line options and arguments.
2. Uses while and getopt function – Look at PDF for more info

## 0.2 Taylor Series Function

1. Implement a function that computes the factorial of a given integer.
2. Implement a function that computes the value of the function at a given point using its definition.
3. Implement a function that computes the Taylor series expansion of the function up to EPSILION.
4. Return the Taylor series expansion as a string.

## 0.3 Main Function

1. Call the input processing function to parse the command line options and the input file.
2. Call the Taylor series expansion function to compute the Taylor series expansion.
3. Call the output generation function to write the output to the console or to a file.
4. Exit the program.

## Algorithms

Since this program is focusing on the Taylor series and the only major difference is the equations. A for loop can represent a summation equation.

```
Taylor Series Template:

for (double i = 0; absolute(term) > EPSILON; i++) {
        if (i == 0) {
            tmp = 1.0;
        } else {
            tmp *= n;
        }
        term = ((1 / tmp) / (2 * i + 1));
        pi += term;
        terms++;
    }
    return sqrt(n) * pi;
```

## Function Descriptions

For each function in your program, you will need to explain your thought process. This means doing the following

- The inputs of every function (even if it's not a parameter)
- The outputs of every function (even if it's not the return value)

- The purpose of each function, a brief description about a sentence long.
- For more complicated functions, include psuedocode that describes how the function works
- For more complicated functions, also include a description of your decision making process; why you chose to use any data structures or control flows that you did.

Do not simply use your code to describe this. This section should be readable to a person with little to no code knowledge.

# Results

Output of the program results in an very close approximation of the true value. Although there is some room for optimization, the program does its intended purpose.

## Error Handling

In order for the program to handle infinite loops and numerical errors. The program checks if the numerical number being added is less than EPSILON.