Assignment 4 – Calc Report Template

Jayden Sangha

CSE 13S – Winter 24

Purpose

Audience for this section: Pretend that you are working in industry, and write this paragraph for your boss. You are answering the basic question, "What does this thing do?". This section can be short. A single paragraph is okay.

Do not just copy the assignment PDF to complete this section, use your own words.

The purpose of this program is to make a scientifc calculator. It works it Reverse Polish Notation and can handle basic math along with cos, sin, and tan.

Questions

Please answer the following questions before you start coding. They will help guide you through the assignment. To make the grader's life easier, please do not remove the questions, and simply put your answers below the text of each question.

- Are there any cases where our sin or cosine formulae can't be used? How can we avoid this?

  They can not be used when the value is not in radians or if it is not between 0 and 2pi.

- What ways (other than changing epsilon) can we use to make our results more accurate? [1]

  We can normalize the input and put it to scale before running it through any algorithms.

- What does it mean to normalize input? What would happen if you didn't?

  Normalizing the input is when you change the input's format to the standard format for the function. Not doing this can lead to issues in computations and machine learning applications due to scaling.

- How would you handle the expression 321+? What should the output be? How can we make this a more understandable RPN expression?

  The function should add 3 then 2 and 1 to the stack. It then adds the top two numbers so the output would be 3. We can make this more understandable by making the expression 2 1 +.

- Does RPN need parenthesis? Why or why not?

RPN does not need parenthesis because it has its own special order of operations that do not require parenthesis.

Testing

List what you will do to test your code. Make sure this is comprehensive. [2] Be sure to test inputs with delays and a wide range of files/characters.

Test when the input is not a radian. Test when the input is outside the 0 to 2pi range.

[1] hint: Use trig identities

[2] This question is a whole lot more vague than it has been the last few assignments. Continue to answer it with the same level of detail and thought.

How to Use the Program

Audience: Write this section for the user of your program. You are answering the basic question, "How do I use this thing?". Don't copy the assignment exactly; explain this in your own words. This section will be longer for a more complicated program and shorter for a less complicated program. You should show how to compile and run your program. You should also describe any optional flags or inputs that your program uses, and what they do.

To use this program you must first compile it by using

```
make calc
```

then

```
./calc
```

to open up the program. You then enter equations in Reverse Polish Notation and it prints the output. You can use Ctr-C to exit the program.

To show "code font" text within a paragraph, you can use `\lstinline{}`, which will look like this: `text`.

For a code block, use `\begin{lstlisting}` and `\end{lstlisting}`, which will look like this:

```
Here is some code in lstlisting.
```

And if you want a box around the code text, then use `\begin{lstlisting}[frame=single]` and `\end{lstlisting}`

which will look like this:

```
Here is some framed code (lstlisting) text.
```

Want to make a footnote? Here's how.[3]

Do you need to cite a reference? You do that by putting the reference in the file `bibtex.bib`, and then you cite your reference like this[**?**][**?**][**?**].

Program Design

Audience: Write this section for someone who will maintain your program. In industry you maintain your own programs, and so your audience could be future you! List the main data structures and the main algorithms. You are answering the basic question, "How is this thing organized so that I can have a chance of fixing it?". This section will be longer for a more complicated program and shorter for a less complicated program.

Pseudocode

Give the reader a top down description of your code! How will you break it down? What features will your code have? How will you implement each function.

```
double sqrt (double x)
if (x<0)
return nan

double old = 0
double new = 0

while abs(old-new) > epsilon
old = new
new = .5 * (old + (x/old))

return new
```

[3]This is my footnote.

```
double abs (double x)
if x < 0
x * -1

return x

double operator_add (double rhs double lhs)
result rhs + lhs
return result


double operator_sub (double rhs double lhs)
result rhs - lhs
return result


double operator_div (double rhs double lhs)
result lhs / rhs
return result


double operator_mul (double rhs double lhs)
result rhs * lhs
return result


int main(argc, argv[])
while 1
char save ptr
bool error
char expr[1024]

print stderr, "> "

if input empty
return 0

if new line
replace with \0



const char *token = strtok_r(expr, " ", \&saveptr)

while token is not 0 and there is no error
if strlen(token) ==1)

case +
if add operator false
print error
error = true

case -
if subtract operator false
print error
error = true

case /
if divide operator returns false
```

```
print error
error = true

case *
if multiply operator returns false
print error
errror= true

case %
if fmod operator fails
print error
error = true

case s
if sin operator fails
print error
error = true

case c
if cosine operator returns false
print error
errror= true

case t
if tangent operator returns false
print error
errror= true

case a
if absolute value operator returns false
print error
errror= true

case r
if squared operator returns false
print error
errror= true

default
print error
error = true



else
print error
error = true
break

token = strtok\_r(NULL, " ", \&saveptr);

if error is false
stack_print()
print result


stack_clear

return 0
```

```
double cos
result = 1
hold = 1

set x to inbetween 0 and 2pi

for (int i = 1; i <= 64; ++i) \{

calcul *= -x * x / (2 * i * (2 * i - 1));

result += hold;

set result between 0 and 2pi

return result

double sin
set x to inbetween 0 and 2pi
y =1
result =x
hold =x
swap = -1

while (1) \{
holder = holder * ((x * x) / ((2 * y) * (2 * y + 1)));
result += swap * holder;
swap = -swap;
y++;
if (Abs(holder) < EPSILON) \{
holder = holder * ((x * x) / ((2 * y) * (2 * y + 1)));
result += swap * holder;
break;

set result between 0 and 2pi
return result

double tan
y = sinx / cosx

return y
```

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 pseudocode 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. **DO NOT JUST PUT THE FUNCTION SIGNATURES HERE. MORE EXPLANATION IS REQUIRED.**

double Abs(double x)

This function takes a double and returns the absolute value of the the double.

double Sqrt(double x)

This function takes a double and returns the square root.

double Sin(double x)

This function takes a double in radians and returns the sine of the input.

double Cos(double x)

This function takes a double in radian and returns the cosine of the input.

double Tan(double x)

This function takes a double in radian and returns the tangent of the input.

bool apply_binary_operator(binary_operator_fun op)

This function takes in an operator and accesses the global stack. It pops the first two elements and calls the op functions with the popped elements. It will then push the result to the stack, but if there are not two elements it will return false.

bool apply_unary_operator(unary_operator_fn op)

This function takes in an operator and accesses the glob stack. It applies the operator to the first elements on the stack and adds the result to the stack.

double operator_add(double lhs, double rhs)

This function returns the sum of the two doubles.

double operator_sub(double lhs, double rhs)

This function returns the differance of the two doubles.

double operator_mul(double lhs, double rhs)

This functions multiplies the two doubles together and returns the result.

double operator_div(double lhs, double rhs)

This functions divides the two doubles and returns the result.

bool parse_double(const char *s, double *d)

This function checks if the string is a valid number if it is then it takes the string as a number and sends it to the location of d.

bool stack_push(double item)

Pushes the item to the top of the stack

bool stack_peek(double *item)

This function copies the top item in the stack to the place the input points too.

bool stack_pop(double *item)

This function stores the top item in the stack to the location indicated by the input.

void stack_clear(void)

This function sets the size of the stack to zero.

void stack_print(void)

This function prints all items in the stack.

Results

Follow the instructions on the pdf to do this. In overleaf, you can drag an image straight into your source code to upload it. You can also look at `https://www.overleaf.com/learn/latex/Inserting_Images`

The main problem I had was I could not get my sine function to work. I initially tried to make it like my cosine function but it kept giving me errors or the wrong result. I realized it was because my function could not handle enough decimal points to give an accurate result. I rewrote my function with a while loop that kept interring through the number until it had an accurate response.

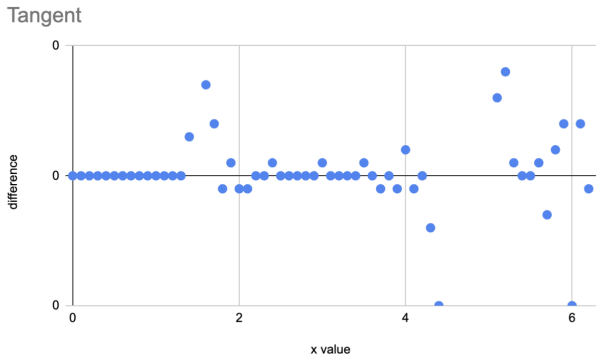

Figure 1: Enter Caption

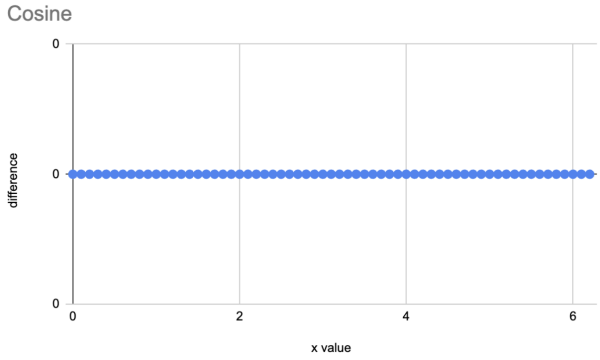

Figure 2: Enter Caption

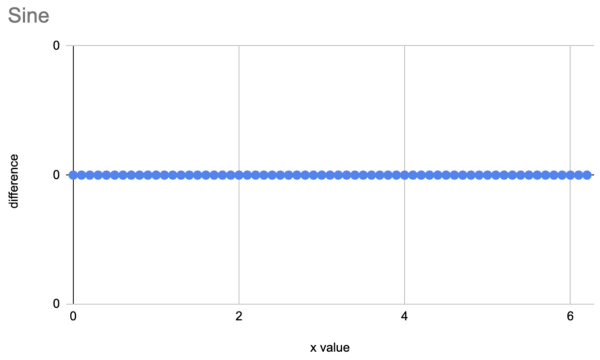

Figure 3: Enter Caption