HW02

Chandler Justice - A023213187

introduction

I decided to use `RustLang` to implement the requirements of this assignment. This is because of its efficiency and memory safe design.

32 bit maceps value

To determine the 32 bit maceps value I wrote the following code:

Maceps float f32

```
fn maceps_float_f32() -> f32{
    let one:f32 = 1.0;
    let mut h:f32 = 1.0;
    let mut appone = one + h;
    let mut error = (appone - one).abs();
    while error > 0.0 {
        h = h / 2.0;
        appone = one + h;
        error = (appone - one).abs();

    }
    h //return statement
}

fn main(){
    let maceps_f32: f32 = maceps_float_f32();
    println!("Maceps float to 32-bits of precision: {maceps_f32}");
}
```

Running this we get the following output:

```
chandler@merci hw02 % cargo run
    Finished dev [unoptimized + debuginfo] target(s) in 0.00s
     Running `target/debug/hw02`
Maceps float to 32-bits of precision: 0.000000059604645
```

64 bit maceps value

The code for determining the 64 bit maceps value is almost identical, except we will declare `f64` data types instead of `f32`:

Maceps float f64

```
fn maceps_float_f64() -> f64{
    let one:f64 = 1.0;
    let mut h:f64 = 1.0;
    let mut appone = one + h;
    let mut error = (appone - one).abs();
    while error > 0.0 {
        h = h / 2.0;
        appone = one + h;
        error = (appone - one).abs();

    }

    h
}

fn main(){
    let maceps_f64: f64 = maceps_float_f64();
    println!("Maceps float to 64-bits of precision: {maceps_f64}");
}
```

Running this we get the following output:

```
chandler@merci hw02 % cargo run
    Finished dev [unoptimized + debuginfo] target(s) in 0.00s
     Running `target/debug/hw02`
Maceps float to 64-bits of precision: 0.00000000000000011102230246251565
```

Norms of single Vectors

I wrote the following three functions to determine the specified norms:

```
fn two_norm(v: &[f64]) -> f64{
    let mut sum: f64 = 0.0;
    for i in 0..v.len(){
        sum += v[i] * v[i];
    }
    sum = sum.sqrt();
    sum
}

fn one_norm(v: &[f64]) -> f64{
    let mut sum: f64 = 0.0;
    for i in 0..v.len(){
        sum += v[i].abs();
    }
    sum
}
```

```
fn inf_norm(v: &[f64]) -> f64 {
    let mut max = 0.0;
    for i in 0..v.len(){
        let abs: f64 = v[i].abs();
        if abs > max{
            max = abs;
        }
    }
    max
}
```

These functions produce the following outputs:

```
    Finished dev [unoptimized + debuginfo] target(s) in 0.80s
     Running `target/debug/hw02`
two norm of [2.4, 4.0, 3.14, 2.0]: 5.968215813792259
one norm of [4.0, 6.7, 4.9, -4.5]: 20.1
infinity norm of [4.0, 6.7, 4.9, -4.5]: 6.7
```

Norms of distance between two vectors

I wrote the following functions to compute the norms of the difference between two vectors:

```
fn two_norm_dist(v1: &[f64], v2: &[f64]) -> f64 {
    if v1.len() != v2.len(){
        panic!("cannot compute two norm as vectors are different sizes.");
    }
    let mut sum: f64 = 0.0;
    for i in 0..v1.len(){
        let distance = v1[i] - v2[i];
        sum += distance * distance;
    }
    sum = sum.sqrt();

    //return
    sum
}

fn one_norm_dist(v1: &[f64], v2: &[f64]) -> f64 {
    if v1.len() != v2.len(){
        panic!("cannot compute two norm as vectors are different sizes.");
    }
    let mut sum: f64 = 0.0;
    for i in 0..v1.len() {
        let distance = v1[i] - v2[i];
        sum += distance.abs();
    }
    //return
    sum
```

```
}

fn inf_norm_dist(v1: &[f64], v2: &[f64]) -> f64 {
    if v1.len() != v2.len(){
        panic!("cannot compute two norm as vectors are different sizes.");
    }
    let mut max: f64 = 0.0;
    for i in 0..v1.len(){
        let distance = (v1[i] - v2[i]).abs();
        if distance > max {
            max = distance;
        }
    }

    max
}
```

These functions produce the following output:

```
    Finished dev [unoptimized + debuginfo] target(s) in 0.25s
     Running `target/debug/hw02`
two norm of [2.4, 4.0, 3.14, 2.0] & [4.0, 6.7, 4.9, -4.5]:
7.429508732076435
one norm of [2.4, 4.0, 3.14, 2.0] & [4.0, 6.7, 4.9, -4.5]: 12.56
inf norm of [2.4, 4.0, 3.14, 2.0] & [4.0, 6.7, 4.9, -4.5]: 6.5
```