14) Show that all of the zeros lie between [-3, 3] for

\[ f(x) = 2x^5 - 13x^3 + 2x - 5 \]

15) List all possible rational roots for

\[ f(x) = 2x^5 - 13x^3 + 2x - 5 \]

A WHALE
is fine too
Functions allow you to reuse pieces of code (either your own or someone else's)

Every function has a return type, specifically the type of object returned

sqrt(2) returns a double, as the number will probably have a fractional part

The “2” is an argument to the sqrt function
Functions

The return statement value must be the same as the return type (or convertible)

```c
int add(int x, int y)
{
    return x+y;
}
```

3 to x, 5 to y... value 8 returned and stored in x
Functions

Function call stack (after returning, start from where the previous function called it)

Overloading - same function name, different arguments (typically similar)

Call-by-reference (not copy)

```c
void changeMe(int &x)
{
    x=2;
}
```

Addresses share

Functions should be minimal
Q1: Write a function that takes in a probability and returns true if the that percent of the time

Q2: Write another function that takes in a probability and an integer, k. This should simulate the probability k times and tell how many times the probability happened.

(See: functionQ.cpp)
Variables exist in the braces where it is declared (in `{ }`).

```c
int x = 3;
int main()
{
    int y = 2;
    if(y < 10)
    {
        int z = 3;
    }
}
```

- x exists anywhere here.
- knows about x and y
- knows x, y and z
Scope

int add(int x, int y);

int main()
{
    int x = add(2, 4);
}

int add(int x, int y)
{
    int z = x+y;
    return z;
}
For files you must first open them:

```cpp
ofstream out;
out.open("output.txt");
```

Then you use “out” instead of “cout” or “cin” depending on if it is an ostream of istream

Also close when done:

```cpp
out.close();
```
File I/O

Can check to see if the program is correctly sending/receiving to/from file:

```cpp
if(out.fail())
{
    exit(1); // non-zero for an error state
}
```

If you want to add to the file instead of replacing it, you have to specify when opening

```cpp
out.open("output.txt", ios::app);
```
End of file (EOF)

When there is nothing left in a file to read, we call it **end of file**

C++ is fairly nice about handling EOF, and you can detect it in 3 ways:

```cpp
while(getline(in, x))
while(in >> x)
while(!in.eof())
```

- **reads from file**
- **does not read from file (just tells if at end)**
Q3: Read all the numbers from "numbers.txt" and put their sum in "sum.txt"

If you cannot read "numbers.txt", put "NaN" into "sum.txt"
(you can get this by doing 0.0/0.0)
(technically the above is -NaN...)

(see: fileQ.cpp)
Arrays store multiple things of the same type

```c
int x[5]; // 5 ints
```

- **Type** , `[]` means array
- **variable name**
- **length of array**

After declaration **any use of [ ]** is interpreted as element indexing

Arrays are memory addresses, shares with functions (cannot call-by-reference)
Multidimensional Arrays

```plaintext
string myArray[4][5];
```

four rows  five columns

Must specify (some parts of) size when using as argument in function
Q4: Write a function that takes two int arrays of length 11 as input. Return true if the first array has more larger numbers when compared to the second element by element:

first = [1, 2, 3, 4], second = [90, 0, 0, 0, 0], then function would return true as first array has 3 larger elements and 1 smaller: 1 < 90, 2 > 0, 3 > 0, 4 > 0
(see: arrayQ.cpp)
Recursion

There are two important parts of recursion:
- A stopping case that ends the recursion
- A reduction case that reduces the problem

Identify the problem sub-structure, then move inputs towards the base case

\[ F_n = F_{n-1} + F_{n-2}, \]
0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ...

You can assume your function works as you want it to (and it will if you do it properly!)
Recursion

Q5: Write a recursive function that keeps asking if the user wants to stop. When the character 'q' is pressed, stop and return how many inputs other than q they entered.

Example input: aabeq
Example output: 4 other numbers

(see: recursionQ.cpp)
C-Strings and strings

c-string uses **null character** to tell when to end

```cpp
char word[] = {'h', 'i', '\0'};
string sameWord = word;
```

(c++) string is a class (which is a type) and is newer and has many functions:
- `find()`
- `substr()`
- `at()` or `[ ]`

Essential for dealing with more than one char at a time
C-Strings and strings

Q6: Write a function that takes a c-string (char array) as input (and its length) and changes it to display half as much (i.e. “cookies” -> “cook” or “coo”) (see: cstringQ.cpp)

Q7: Make a word game that repeatedly reads in words until the user repeats a word they have already entered. At this point tell the user they have lost (see: wordGame.cpp)