Write Function sqrt

//return the square root of the parameter value

double sqrt(double a){
  /* The square root of a when multiply by itself, the result is a
     - square root is a number that divides a,
     where the divisor and quotient is same or very close
     Ex: 4/2 = 2, 9/3 = 3, 6/2.44949 = 2.44948
  */
  double div = 2;
  double quo = a / div;
  while (!isClose(div, quo)) {
    div = (div + quo) / 2; // 6 = 2 * 3
    quo = a / div;
  }
  return div;
}

bool isClose(double a, double b){
  return abs(a - b) < 0.0001;
}
#include <iostream>
Using namespace std;

double max(double a, double b, double c);
double max(double a, double b);

int main() {
  double a, b, c;
  cout << "Enter 3 values: ";
  cin >> a >> b >> c;
  cout << max(a, c, b);
  cout << max(a + 1, b - 1);
  return 0;
}

double max(double a, double b, double c) {
  return max(max(a, b), c);
}

double max(double a, double b) {
  if (a > b) return a;
  return b;
}
#include <iostream>

using namespace std;

// function swap two values
void swap(int a, int b);

int main()
{
    int x = 10, y = 20;
    cout << "Start: x- " << x << " y- " << y << endl;
    swap(x, y);
    cout << "End: x- " << x << " y- " << y << endl;
    return 0;
}

void swap(int a, int b)
{
    cout << "In swap: a- " << a << " b- " << b << endl;
    int temp = a;
    a = b;
    b = temp;
    cout << "In swap: a- " << a << " b- " << b << endl;
}
Swap Function doesn’t work?

Any variable values we change in the function has nothing to do with the values we passed in.

These are different variables.

```c
void swap(int a, int b){
    int temp = a;
    a = b;
    b = temp;
}
int main(){
    int x = 10, y = 20;
    swap(x, y);
    return 0;
}
```

<table>
<thead>
<tr>
<th>main:x</th>
<th>main:y</th>
<th>swap:a</th>
<th>swap:b</th>
<th>swap:temp</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>20</td>
<td>20</td>
<td>10</td>
<td>10</td>
</tr>
</tbody>
</table>
Pass by Value vs Pass by Reference

- What we see in the previous example, for each function call, we create new variables for each parameter, and copy of the argument values into these parameters. This is called Pass by Value.
- Any primitive data types (the ones we’ve learn so far) are pass by value by default.

- We may order the compiler to not create separate values for each parameter.

- Instead, make them reference the SAME value. This is called Pass by Reference.
Pass by Reference

By putting an ampersand sign & before the parameter, now the argument is the parameter.

These are same variables.

```c
void swap(int &a, int &b){
    int temp = a;
    a = b;
    b = temp;
}
int main(){
    int x = 10, y = 20;
    swap(x, y);
    return 0;
}
```

---

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>main:x</td>
<td>20</td>
</tr>
<tr>
<td>swap:a</td>
<td>10</td>
</tr>
<tr>
<td>main:y</td>
<td>10</td>
</tr>
<tr>
<td>swap:temp</td>
<td>10</td>
</tr>
</tbody>
</table>
Pass by Reference

- When calling a function, if it’s parameter is pass by reference, when we can’t pass a hard coded value or evaluated value as an argument to it. It only takes a variable, which the parameter shares.

```cpp
void swap(int &a, int &b);
int main(){
    int x = 10, y = 20;
    swap(x, y);
    swap(2, 3);
    swap(x - 1, y);
    return 0;
}
```