Function Call Trace

```
public class Newton {
   public static double sqrt(double c) {
      double epsilon = 1e-15;
      if (c < 0) return Double.NaN;
      double t = c;
      while (Math.abs(t - c/t) > epsilon * t)
         t = (c/t + t) / 2.0;
      return t;
   }
   public static void main(String[] args) {
      double[] a = new double[args.length];
      for (int i = 0; i < args.length; i++)
         a[i] = Double.parseDouble(args[i]);
      for (int i = 0; i < a.length; i++)
         System.out.println(sqrt(a[i]));
   }
}
```

1

Function Call Trace

```
public class Newton {
   public static double sqrt(double c) {
      double epsilon = 1e-15;
      if (c < 0) return Double.NaN;
      double t = c;
      while (Math.abs(t - c/t) > epsilon * t)
         t = (c/t + t) / 2.0;
      return t;
   }
   public static void main(String[] args) {
      double[] a = new double[args.length];
      for (int i = 0; i < args.length; i++)
         a[i] = Double.parseDouble(args[i]);
      for (int i = 0; i < a.length; i++)
         System.out.println(sqrt(a[i]));
   }
}
```

```
% java Newton 1 2 3
```

2

Function Call Trace

```
public class Newton {
   public static double sqrt(double c) {
      double epsilon = 1e-15;
      if (c < 0) return Double.NaN;
      double t = c;
      while (Math.abs(t - c/t) > epsilon * t)
         t = (c/t + t) / 2.0;
      return t;
   }
   public static void main(String[] args) {
      double[] a = new double[args.length];
      for (int i = 0; i < args.length; i++)
         a[i] = Double.parseDouble(args[i]);
      for (int i = 0; i < a.length; i++)
         System.out.println(sqrt(a[i]));
   }
}
```

{ "1", "2", "3" }

```
% java Newton 1 2 3
```

args[]

{ "1", "2", "3" }

3

Function Call Trace

```
public class Newton {
   public static double sqrt(double c) {
      double epsilon = 1e-15;
      if (c < 0) return Double.NaN;
      double t = c;
      while (Math.abs(t - c/t) > epsilon * t)
         t = (c/t + t) / 2.0;
      return t;
   }
   public static void main(String[] args) {
      double[] a = new double[args.length];
      for (int i = 0; i < args.length; i++)
         a[i] = Double.parseDouble(args[i]);
      for (int i = 0; i < a.length; i++)
         System.out.println(sqrt(a[i]));
   }
}
```

```
% java Newton 1 2 3
```

args[]

{ "1", "2", "3" }

a[]

{ 0.0, 0.0, 0.0 }

4

Function Call Trace

```
public class Newton {
   public static double sqrt(double c) {
      double epsilon = 1e-15;
      if (c < 0) return Double.NaN;
      double t = c;
      while (Math.abs(t - c/t) > epsilon * t)
         t = (c/t + t) / 2.0;
      return t;
   }
   public static void main(String[] args) {
      double[] a = new double[args.length];
      for (int i = 0; i < args.length; i++)
         a[i] = Double.parseDouble(args[i]);
      for (int i = 0; i < a.length; i++)
         System.out.println(sqrt(a[i]));
   }
}
```

```
% java Newton 1 2 3
```

args[]

{ "1", "2", "3" }

a[]

{ 0.0, 0.0, 0.0 }

i

0

5

Function Call Trace

```
public class Newton {
   public static double sqrt(double c) {
      double epsilon = 1e-15;
      if (c < 0) return Double.NaN;
      double t = c;
      while (Math.abs(t - c/t) > epsilon * t)
         t = (c/t + t) / 2.0;
      return t;
   }
   public static void main(String[] args) {
      double[] a = new double[args.length];
      for (int i = 0; i < args.length; i++)
         a[i] = Double.parseDouble(args[i]);
      for (int i = 0; i < a.length; i++)
         System.out.println(sqrt(a[i]));
   }
}
```

```
% java Newton 1 2 3
```

args[]

{ "1", "2", "3" }

a[]

{ 1.0, 0.0, 0.0 }

i

0

6

Function Call Trace

```
public class Newton {
   public static double sqrt(double c) {
      double epsilon = 1e-15;
      if (c < 0) return Double.NaN;
      double t = c;
      while (Math.abs(t - c/t) > epsilon * t)
         t = (c/t + t) / 2.0;
      return t;
   }
   public static void main(String[] args) {
      double[] a = new double[args.length];
      for (int i = 0; i < args.length; i++)
         a[i] = Double.parseDouble(args[i]);
      for (int i = 0; i < a.length; i++)
         System.out.println(sqrt(a[i]));
   }
}
```

```
% java Newton 1 2 3
```

args[]: { "1", "2", "3" }

a[]: { 1.0, 0.0, 0.0 }

i: 1

7

Function Call Trace

```
public class Newton {
   public static double sqrt(double c) {
      double epsilon = 1e-15;
      if (c < 0) return Double.NaN;
      double t = c;
      while (Math.abs(t - c/t) > epsilon * t)
         t = (c/t + t) / 2.0;
      return t;
   }
   public static void main(String[] args) {
      double[] a = new double[args.length];
      for (int i = 0; i < args.length; i++)
         a[i] = Double.parseDouble(args[i]);
      for (int i = 0; i < a.length; i++)
         System.out.println(sqrt(a[i]));
   }
}
```

```
% java Newton 1 2 3
```

args[]: { "1", "2", "3" }

a[]: { 1.0, 2.0, 0.0 }

i: 1

8

Function Call Trace

```
public class Newton {
   public static double sqrt(double c) {
      double epsilon = 1e-15;
      if (c < 0) return Double.NaN;
      double t = c;
      while (Math.abs(t - c/t) > epsilon * t)
         t = (c/t + t) / 2.0;
      return t;
   }
   public static void main(String[] args) {
      double[] a = new double[args.length];
      for (int i = 0; i < args.length; i++)
         a[i] = Double.parseDouble(args[i]);
      for (int i = 0; i < a.length; i++)
         System.out.println(sqrt(a[i]));
   }
}
```

```
% java Newton 1 2 3
```

args[]: { "1", "2", "3" }

a[]: { 1.0, 2.0, 0.0 }

i: 2

9

Function Call Trace

```
public class Newton {
   public static double sqrt(double c) {
      double epsilon = 1e-15;
      if (c < 0) return Double.NaN;
      double t = c;
      while (Math.abs(t - c/t) > epsilon * t)
         t = (c/t + t) / 2.0;
      return t;
   }
   public static void main(String[] args) {
      double[] a = new double[args.length];
      for (int i = 0; i < args.length; i++)
         a[i] = Double.parseDouble(args[i]);
      for (int i = 0; i < a.length; i++)
         System.out.println(sqrt(a[i]));
   }
}
```

```
% java Newton 1 2 3
```

args[]: { "1", "2", "3" }

a[]: { 1.0, 2.0, 3.0 }

i: 2

10

Function Call Trace

```
public class Newton {
   public static double sqrt(double c) {
      double epsilon = 1e-15;
      if (c < 0) return Double.NaN;
      double t = c;
      while (Math.abs(t - c/t) > epsilon * t)
         t = (c/t + t) / 2.0;
      return t;
   }
   public static void main(String[] args) {
      double[] a = new double[args.length];
      for (int i = 0; i < args.length; i++)
         a[i] = Double.parseDouble(args[i]);
      for (int i = 0; i < a.length; i++)
         System.out.println(sqrt(a[i]));
   }
}
```

```
% java Newton 1 2 3
```

args[]: { "1", "2", "3" }

a[]: { 1.0, 2.0, 3.0 }

i: 3 (goes out of scope)

11

Function Call Trace

```
public class Newton {
   public static double sqrt(double c) {
      double epsilon = 1e-15;
      if (c < 0) return Double.NaN;
      double t = c;
      while (Math.abs(t - c/t) > epsilon * t)
         t = (c/t + t) / 2.0;
      return t;
   }
   public static void main(String[] args) {
      double[] a = new double[args.length];
      for (int i = 0; i < args.length; i++)
         a[i] = Double.parseDouble(args[i]);
      for (int i = 0; i < a.length; i++)
         System.out.println(sqrt(a[i]));
   }
}
```

```
% java Newton 1 2 3
```

args[]: { "1", "2", "3" }

a[]: { 1.0, 2.0, 3.0 }

i: 0

12

Function Call Trace

```
public class Newton {
   public static double sqrt(double c) {
      double epsilon = 1e-15;
      if (c < 0) return Double.NaN;
      double t = c;
      while (Math.abs(t - c/t) > epsilon * t)
         t = (c/t + t) / 2.0;
      return t;
   }
   public static void main(String[] args) {
      double[] a = new double[args.length];
      for (int i = 0; i < args.length; i++)
         a[i] = Double.parseDouble(args[i]);
      for (int i = 0; i < a.length; i++)
         System.out.println(sqrt(a[i]));
   }
}
```

1.0 ← c

1.0 ← sqrt(a[i])

args[]: { "1", "2", "3" }

a[]: { 1.0, 2.0, 3.0 }

i: 0

```
% java Newton 1 2 3
```

13

Function Call Trace

```
public class Newton {
   public static double sqrt(double c) {
      double epsilon = 1e-15;
      if (c < 0) return Double.NaN;
      double t = c;
      while (Math.abs(t - c/t) > epsilon * t)
         t = (c/t + t) / 2.0;
      return t;
   }
   public static void main(String[] args) {
      double[] a = new double[args.length];
      for (int i = 0; i < args.length; i++)
         a[i] = Double.parseDouble(args[i]);
      for (int i = 0; i < a.length; i++)
         System.out.println(sqrt(a[i]));
   }
}
```

1.0 ← c

1.0 ← sqrt(a[i])

c: 1.0

args[]: { "1", "2", "3" }

a[]: { 1.0, 2.0, 3.0 }

i: 0

```
% java Newton 1 2 3
```

14

Function Call Trace

```
public class Newton {
   public static double sqrt(double c) {
      double epsilon = 1e-15;
      if (c < 0) return Double.NaN;
      double t = c;
      while (Math.abs(t - c/t) > epsilon * t)
         t = (c/t + t) / 2.0;
      return t;
   }
   public static void main(String[] args) {
      double[] a = new double[args.length];
      for (int i = 0; i < args.length; i++)
         a[i] = Double.parseDouble(args[i]);
      for (int i = 0; i < a.length; i++)
         System.out.println(sqrt(a[i]));
   }
}
```

c: 1.0

args[]: { "1", "2", "3" }

a[]: { 1.0, 2.0, 3.0 }

i: 0

```
% java Newton 1 2 3
```

15

Function Call Trace

```
public class Newton {
   public static double sqrt(double c) {
      double epsilon = 1e-15;
      if (c < 0) return Double.NaN;
      double t = c;
      while (Math.abs(t - c/t) > epsilon * t)
         t = (c/t + t) / 2.0;
      return t;
   }
   public static void main(String[] args) {
      double[] a = new double[args.length];
      for (int i = 0; i < args.length; i++)
         a[i] = Double.parseDouble(args[i]);
      for (int i = 0; i < a.length; i++)
         System.out.println(sqrt(a[i]));
   }
}
```

c: 1.0

args[]: { "1", "2", "3" }

a[]: { 1.0, 2.0, 3.0 }

i: 0

```
% java Newton 1 2 3
```

16

Function Call Trace

```
public class Newton {
   public static double sqrt(double c) {
      double epsilon = 1e-15;
      if (c < 0) return Double.NaN;
      double t = c;
      while (Math.abs(t - c/t) > epsilon * t)
         t = (c/t + t) / 2.0;
      return t;
   }
   public static void main(String[] args) {
      double[] a = new double[args.length];
      for (int i = 0; i < args.length; i++)
         a[i] = Double.parseDouble(args[i]);
      for (int i = 0; i < a.length; i++)
         System.out.println(sqrt(a[i]));
   }
}
```

c: 1.0 t: 1.0

args[]: { "1", "2", "3" }

a[]: { 1.0, 2.0, 3.0 }

i: 0

```
% java Newton 1 2 3
```

17

Function Call Trace

```
public class Newton {
   public static double sqrt(double c) {
      double epsilon = 1e-15;
      if (c < 0) return Double.NaN;
      double t = c;
      while (Math.abs(t - c/t) > epsilon * t)
         t = (c/t + t) / 2.0;
      return t;
   }
   public static void main(String[] args) {
      double[] a = new double[args.length];
      for (int i = 0; i < args.length; i++)
         a[i] = Double.parseDouble(args[i]);
      for (int i = 0; i < a.length; i++)
         System.out.println(sqrt(a[i]));
   }
}
```

c: 1.0 t: 1.0

args[]: { "1", "2", "3" }

a[]: { 1.0, 2.0, 3.0 }

i: 0

```
% java Newton 1 2 3
```

18

Function Call Trace

```
public class Newton {
   public static double sqrt(double c) {
      double epsilon = 1e-15;
      if (c < 0) return Double.NaN;
      double t = c;
      while (Math.abs(t - c/t) > epsilon * t)
         t = (c/t + t) / 2.0;
      return t;
   }
   public static void main(String[] args) {
      double[] a = new double[args.length];
      for (int i = 0; i < args.length; i++)
         a[i] = Double.parseDouble(args[i]);
      for (int i = 0; i < a.length; i++)
         System.out.println(sqrt(a[i]));
   }
}
```

c: 1.0 t: 1.0

args[]: { "1", "2", "3" }

a[]: { 1.0, 2.0, 3.0 }

i: 0

```
% java Newton 1 2 3
```

19

Function Call Trace

```
public class Newton {
   public static double sqrt(double c) {
      double epsilon = 1e-15;
      if (c < 0) return Double.NaN;
      double t = c;
      while (Math.abs(t - c/t) > epsilon * t)
         t = (c/t + t) / 2.0;
      return t;
   }
   public static void main(String[] args) {
      double[] a = new double[args.length];
      for (int i = 0; i < args.length; i++)
         a[i] = Double.parseDouble(args[i]);
      for (int i = 0; i < a.length; i++)
         System.out.println(sqrt(a[i]));
   }
}
```

args[]: { "1", "2", "3" }

a[]: { 1.0, 2.0, 3.0 }

i: 0

```
% java Newton 1 2 3
1.0
```

20

Function Call Trace

```
public class Newton {
   public static double sqrt(double c) {
      double epsilon = 1e-15;
      if (c < 0) return Double.NaN;
      double t = c;
      while (Math.abs(t - c/t) > epsilon * t)
         t = (c/t + t) / 2.0;
      return t;
   }
   public static void main(String[] args) {
      double[] a = new double[args.length];
      for (int i = 0; i < args.length; i++)
         a[i] = Double.parseDouble(args[i]);
      for (int i = 0; i < a.length; i++)
         System.out.println(sqrt(a[i]));
   }
}
```

args[]: { "1", "2", "3" }

a[]: { 1.0, 2.0, 3.0 }

i: 1

```
% java Newton 1 2 3
1.0
```

21

Function Call Trace

```
public class Newton {
   public static double sqrt(double c) {
      double epsilon = 1e-15;
      if (c < 0) return Double.NaN;
      double t = c;
      while (Math.abs(t - c/t) > epsilon * t)
         t = (c/t + t) / 2.0;
      return t;
   }
   public static void main(String[] args) {
      double[] a = new double[args.length];
      for (int i = 0; i < args.length; i++)
         a[i] = Double.parseDouble(args[i]);
      for (int i = 0; i < a.length; i++)
         System.out.println(sqrt(a[i]));
   }
}
```

c: 2.0

args[]: { "1", "2", "3" }

a[]: { 1.0, 2.0, 3.0 }

i: 1

```
% java Newton 1 2 3
1.0
```

22

Function Call Trace

```
public class Newton {
   public static double sqrt(double c) {
      double epsilon = 1e-15;
      if (c < 0) return Double.NaN;
      double t = c;
      while (Math.abs(t - c/t) > epsilon * t)
         t = (c/t + t) / 2.0;
      return t;
   }
   public static void main(String[] args) {
      double[] a = new double[args.length];
      for (int i = 0; i < args.length; i++)
         a[i] = Double.parseDouble(args[i]);
      for (int i = 0; i < a.length; i++)
         System.out.println(sqrt(a[i]));
   }
}
```

c: 2.0

args[]: { "1", "2", "3" }

a[]: { 1.0, 2.0, 3.0 }

i: 1

```
% java Newton 1 2 3
1.0
```

23

Function Call Trace

```
public class Newton {
   public static double sqrt(double c) {
      double epsilon = 1e-15;
      if (c < 0) return Double.NaN;
      double t = c;
      while (Math.abs(t - c/t) > epsilon * t)
         t = (c/t + t) / 2.0;
      return t;
   }
   public static void main(String[] args) {
      double[] a = new double[args.length];
      for (int i = 0; i < args.length; i++)
         a[i] = Double.parseDouble(args[i]);
      for (int i = 0; i < a.length; i++)
         System.out.println(sqrt(a[i]));
   }
}
```

c: 2.0

args[]: { "1", "2", "3" }

a[]: { 1.0, 2.0, 3.0 }

i: 1

```
% java Newton 1 2 3
1.0
```

24

Function Call Trace

```
public class Newton {
   public static double sqrt(double c) {
      double epsilon = 1e-15;
      if (c < 0) return Double.NaN;
      double t = c;
      while (Math.abs(t - c/t) > epsilon * t)
         t = (c/t + t) / 2.0;
      return t;
   }
   public static void main(String[] args) {
      double[] a = new double[args.length];
      for (int i = 0; i < args.length; i++)
         a[i] = Double.parseDouble(args[i]);
      for (int i = 0; i < a.length; i++)
         System.out.println(sqrt(a[i]));
   }
}
```

```
% java Newton 1 2 3
1.0
```

c: 2.0 t: 2.0

args[]: { "1", "2", "3" }

a[]: { 1.0, 2.0, 3.0 }

i: 1

25

Function Call Trace

```
public class Newton {
   public static double sqrt(double c) {
      double epsilon = 1e-15;
      if (c < 0) return Double.NaN;
      double t = c;
      while (Math.abs(t - c/t) > epsilon * t)
         t = (c/t + t) / 2.0;
      return t;
   }
   public static void main(String[] args) {
      double[] a = new double[args.length];
      for (int i = 0; i < args.length; i++)
         a[i] = Double.parseDouble(args[i]);
      for (int i = 0; i < a.length; i++)
         System.out.println(sqrt(a[i]));
   }
}
```

```
% java Newton 1 2 3
1.0
```

c: 2.0 t: 2.0

args[]: { "1", "2", "3" }

a[]: { 1.0, 2.0, 3.0 }

i: 1

26

Function Call Trace

```
public class Newton {
   public static double sqrt(double c) {
      double epsilon = 1e-15;
      if (c < 0) return Double.NaN;
      double t = c;
      while (Math.abs(t - c/t) > epsilon * t)
         t = (c/t + t) / 2.0;
      return t;
   }
   public static void main(String[] args) {
      double[] a = new double[args.length];
      for (int i = 0; i < args.length; i++)
         a[i] = Double.parseDouble(args[i]);
      for (int i = 0; i < a.length; i++)
         System.out.println(sqrt(a[i]));
   }
}
```

```
% java Newton 1 2 3
1.0
```

c: 2.0 t: 1.41

args[]: { "1", "2", "3" }

a[]: { 1.0, 2.0, 3.0 }

i: 1

27

Function Call Trace

```
public class Newton {
   public static double sqrt(double c) {
      double epsilon = 1e-15;
      if (c < 0) return Double.NaN;
      double t = c;
      while (Math.abs(t - c/t) > epsilon * t)
         t = (c/t + t) / 2.0;
      return t;
   }
   public static void main(String[] args) {
      double[] a = new double[args.length];
      for (int i = 0; i < args.length; i++)
         a[i] = Double.parseDouble(args[i]);
      for (int i = 0; i < a.length; i++)
         System.out.println(sqrt(a[i]));
   }
}
```

```
% java Newton 1 2 3
1.0
```

c: 2.0 t: 1.41

args[]: { "1", "2", "3" }

a[]: { 1.0, 2.0, 3.0 }

i: 1

28

Function Call Trace

```
public class Newton {
   public static double sqrt(double c) {
      double epsilon = 1e-15;
      if (c < 0) return Double.NaN;
      double t = c;
      while (Math.abs(t - c/t) > epsilon * t)
         t = (c/t + t) / 2.0;
      return t;
   }
   public static void main(String[] args) {
      double[] a = new double[args.length];
      for (int i = 0; i < args.length; i++)
         a[i] = Double.parseDouble(args[i]);
      for (int i = 0; i < a.length; i++)
         System.out.println(sqrt(a[i]));
   }
}
```

```
% java Newton 1 2 3
1.0
```

c: 2.0 t: 1.41

1.414213562373095

1.414213562373095

args[]: { "1", "2", "3" }

a[]: { 1.0, 2.0, 3.0 }

i: 1

29

Function Call Trace

```
public class Newton {
   public static double sqrt(double c) {
      double epsilon = 1e-15;
      if (c < 0) return Double.NaN;
      double t = c;
      while (Math.abs(t - c/t) > epsilon * t)
         t = (c/t + t) / 2.0;
      return t;
   }
   public static void main(String[] args) {
      double[] a = new double[args.length];
      for (int i = 0; i < args.length; i++)
         a[i] = Double.parseDouble(args[i]);
      for (int i = 0; i < a.length; i++)
         System.out.println(sqrt(a[i]));
   }
}
```

```
% java Newton 1 2 3
1.0
1.414213562373095
```

1.414213562373095

args[]: { "1", "2", "3" }

a[]: { 1.0, 2.0, 3.0 }

i: 1

30

Function Call Trace

```
public class Newton {
   public static double sqrt(double c) {
      double epsilon = 1e-15;
      if (c < 0) return Double.NaN;
      double t = c;
      while (Math.abs(t - c/t) > epsilon * t)
         t = (c/t + t) / 2.0;
      return t;
   }
   public static void main(String[] args) {
      double[] a = new double[args.length];
      for (int i = 0; i < args.length; i++)
         a[i] = Double.parseDouble(args[i]);
      for (int i = 0; i < a.length; i++)
         System.out.println(sqrt(a[i]));
   }
}
```

```
% java Newton 1 2 3
1.0
1.414213562373095
```

args[] { "1", "2", "3" }

a[] { 1.0, 2.0, 3.0 }

i 2

31

Function Call Trace

```
public class Newton {
   public static double sqrt(double c) {
      double epsilon = 1e-15;
      if (c < 0) return Double.NaN;
      double t = c;
      while (Math.abs(t - c/t) > epsilon * t)
         t = (c/t + t) / 2.0;
      return t;
   }
   public static void main(String[] args) {
      double[] a = new double[args.length];
      for (int i = 0; i < args.length; i++)
         a[i] = Double.parseDouble(args[i]);
      for (int i = 0; i < a.length; i++)
         System.out.println(sqrt(a[i]));
   }
}
```

3.0 ←

3.0

```
% java Newton 1 2 3
1.0
1.414213562373095
```

c 3.0

args[] { "1", "2", "3" }

a[] { 1.0, 2.0, 3.0 }

i 2

32

Function Call Trace

```
public class Newton {
   public static double sqrt(double c) {
      double epsilon = 1e-15;
      if (c < 0) return Double.NaN;
      double t = c;
      while (Math.abs(t - c/t) > epsilon * t)
         t = (c/t + t) / 2.0;
      return t;
   }
   public static void main(String[] args) {
      double[] a = new double[args.length];
      for (int i = 0; i < args.length; i++)
         a[i] = Double.parseDouble(args[i]);
      for (int i = 0; i < a.length; i++)
         System.out.println(sqrt(a[i]));
   }
}
```

1.7320508075688772 ←

1.7320508075688772

```
% java Newton 1 2 3
1.0
1.414213562373095
```

c 3.0

args[] { "1", "2", "3" }

a[] { 1.0, 2.0, 3.0 }

i 2

33

Function Call Trace

```
public class Newton {
   public static double sqrt(double c) {
      double epsilon = 1e-15;
      if (c < 0) return Double.NaN;
      double t = c;
      while (Math.abs(t - c/t) > epsilon * t)
         t = (c/t + t) / 2.0;
      return t;
   }
   public static void main(String[] args) {
      double[] a = new double[args.length];
      for (int i = 0; i < args.length; i++)
         a[i] = Double.parseDouble(args[i]);
      for (int i = 0; i < a.length; i++)
         System.out.println(sqrt(a[i]));
   }
}
```

```
% java Newton 1 2 3
1.0
1.414213562373095
1.7320508075688772
```

args[] { "1", "2", "3" }

a[] { 1.0, 2.0, 3.0 }

i 2

34

Function Call Trace

```
public class Newton {
   public static double sqrt(double c) {
      double epsilon = 1e-15;
      if (c < 0) return Double.NaN;
      double t = c;
      while (Math.abs(t - c/t) > epsilon * t)
         t = (c/t + t) / 2.0;
      return t;
   }
   public static void main(String[] args) {
      double[] a = new double[args.length];
      for (int i = 0; i < args.length; i++)
         a[i] = Double.parseDouble(args[i]);
      for (int i = 0; i < a.length; i++)
         System.out.println(sqrt(a[i]));
   }
}
```

```
% java Newton 1 2 3
1.0
1.414213562373095
1.7320508075688772
```

args[] { "1", "2", "3" }

a[] { 1.0, 2.0, 3.0 }

i 3

35

Function Call Trace

```
public class Newton {
   public static double sqrt(double c) {
      double epsilon = 1e-15;
      if (c < 0) return Double.NaN;
      double t = c;
      while (Math.abs(t - c/t) > epsilon * t)
         t = (c/t + t) / 2.0;
      return t;
   }
   public static void main(String[] args) {
      double[] a = new double[args.length];
      for (int i = 0; i < args.length; i++)
         a[i] = Double.parseDouble(args[i]);
      for (int i = 0; i < a.length; i++)
         System.out.println(sqrt(a[i]));
   }
}
```

```
% java Newton 1 2 3
1.0
1.414213562373095
1.7320508075688772
```

args[] { "1", "2", "3" }

a[] { 1.0, 2.0, 3.0 }

36