University of British Columbia  
CPSC 111, Intro to Computation  
Alan J. Hu

Review while Loops  
for Loops  
Nested Loops  
(Practice with if statements and javadoc)

Lecture 12
Some slides borrowed from Kurt Eiselt, Tamara Munzner, and Beth Simon.

Readings
This Week: Ch 6 (Ch 7 in old 2nd ed).
(Reminder: Readings are absolutely vital for learning this stuff!)

Labs and Tutorials
This week is Lab #6.
(Any comments on TA Elan Dubrofsky, please email me.)

Midterms – Save the Dates!
■ Midterm #1 is 5:30-6:30pm on February 10 (Tuesday) in Woodward IRC 2
■ Midterm #2 is 6-7pm on March 11 (Wednesday) in Woodward IRC 2

■ If you have an unavoidable conflict for the March 11 exam, you must email me (with your name, student ID, and description of the time conflict) by Wednesday, March 4!

Programming Assignment 2
■ Assignment 2 is up on WebCT!  
■ Click on the “Assignments” icon.  
■ Due at NOON, March 10 (Tuesday), via electronic hand in.  
■ Start early! (I really mean it!)  
■ There is some Eclipse setup.

Learning Goals
By the end of the next several lectures you will be able to...
■ Write programs that make decisions ("conditionals", aka if statements) and repeat computations ("iteration", while loops, for loops)
Learning Goals
By the end of class today you will be able to...

- Write simple while loops and trace their execution, with more confidence!
- Write for loops and trace their execution.
- Start nesting loops.
- Write nested if statements with more confidence
- Use javadoc to generate nicely formatted API webpages.

Review: while Statement

```
while ( boolean expression )

  body
```

- Simplest form of loop in Java
- Body of loop can be
  - single statement
  - whole block of many statements in curly braces
- Meaning is kind of like English:
  “While you’re still hungry, have some more fries.”
  ```
  while ( still hungry ) {
    eat fries;
  }
```

Loan Amortization

- Suppose you have a $100,000 loan at 5% interest, and you make an annual payment of $6000. How many years will it take you to pay the loan off?

Initially: You owe $100,000
After 1 year:
- Interest accrued: $100,000 * 0.05 = $5000
- Payment: $6000
- You owe $100,000 + $5000 - $6000 = $99,000

After 2 years:
- Interest accrued: $99,000 * 0.05 = $4950
- Payment: $6000
- You owe $99,000 + $4950 - $6000 = $97,950

Are you out of debt yet?

Loan Amortization

- After 2 years:
  - Interest accrued: $99,000 * 0.05 = $4950
  - Payment: $6000
  - You owe $99,000 + $4950 - $6000 = $97,950
- After 3 years…

  Messy computation. Perfect for a computer! Let’s do it!

  (Oh yeah, I’ve written a Loan class…)
Loan Amortization
- How about printing a table showing the balance after each year for the next 3 years? The next 25 years?

Java Shorthand: for Loops
- Loops like this are really common:

```java
int year = 0;
while (year < term) {
    myLoan.addInterest();
    myLoan.makePayment(annualPayment);
    System.out.println("Balance after …");
    year++;
}
```

Java Shorthand: for Loops
- Four main parts of a loop: 1. Initialize
  ```java
  int year = 0;
  ```

  ```java
  while (year < term) {
      myLoan.addInterest();
      myLoan.makePayment(annualPayment);
      System.out.println("Balance after …");
      year++;
  }
  ```

Java Shorthand: for Loops
- Four main parts of a loop: 2. Loop Test
  ```java
  int year = 0;
  while (year < term) {
      myLoan.addInterest();
      myLoan.makePayment(annualPayment);
      System.out.println("Balance after …");
      year++;
  }
  ```

Java Shorthand: for Loops
- Four main parts of a loop: 3. Compute stuff in the loop.
  ```java
  int year = 0;
  while (year < term) {
      myLoan.addInterest();
      myLoan.makePayment(annualPayment);
      System.out.println("Balance after …");
      year++;
  }
  ```

Java Shorthand: for Loops
- Four main parts of a loop: 4. Update loop variable.
  ```java
  int year = 0;
  while (year < term) {
      myLoan.addInterest();
      myLoan.makePayment(annualPayment);
      System.out.println("Balance after …");
      year++;
  }
  ```
Java Shorthand: for Loops

Java has a for loop statement to make this easy:

```java
int year = 0;
while (year < term) {
    myLoan.addInterest();
    myLoan.makePayment(annualPayment);
    System.out.println("Balance after …");
    year++;
}
```

Java Shorthand: for Loops

Java has a for loop statement to make this easy:

```java
for (int year = 0; year < term; year++) {
    myLoan.addInterest();
    myLoan.makePayment(annualPayment);
    System.out.println("Balance after …");
    year++;
}
```

Java Shorthand: for Loops

Java has a for loop statement to make this easy:

```java
for (int year = 0; year < term; year++) {
    myLoan.addInterest();
    myLoan.makePayment(annualPayment);
    System.out.println("Balance after …");
    year++;
}
```

Java Shorthand: for Loops

Java has a for loop statement to make this easy:

```java
for (int year = 0; year < term; year++) {
    myLoan.addInterest();
    myLoan.makePayment(annualPayment);
    System.out.println("Balance after …");
    year++;
}
```

Java Shorthand: for Loops

1. Initialize

```java
for (int year = 0; year < term; year++) {
    myLoan.addInterest();
    myLoan.makePayment(annualPayment);
    System.out.println("Balance after …");
}
```

Java Shorthand: for Loops

2. Loop Test

```java
for (int year = 0; year < term; year++) {
    myLoan.addInterest();
    myLoan.makePayment(annualPayment);
    System.out.println("Balance after …");
}
```
Java Shorthand: for Loops

3. Loop Body

```java
for (int year = 0; year < term; year++) {
    myLoan.addInterest();
    myLoan.makePayment(annualPayment);
    System.out.println("Balance after …");
}
```

4. Update Loop Variable

```java
for (int year = 0; year < term; year++) {
    myLoan.addInterest();
    myLoan.makePayment(annualPayment);
    System.out.println("Balance after …");
}
```

for Versus while Statement

- Anything that can be done with one type of loop can be done with another
- for statement convenient when
  - loop should be executed specific number of times
  - number can be determined before loop starts
- while statement convenient when
  - don’t know yet how many times to execute loop body
  - but can check if it’s time to end loop as you go

Nested Loops

- A loop is just a big statement, so we can put loops inside other loops, or loops inside if statements, etc.

Nested Loops

- Very simple for loop

```java
public class SimpleLoop {
    public static void main (String[] args)
    {
        for (int i = 1; i <= 3; i++)
        {
            System.out.println(i);
        }
    }
}
```

- What does it do?

```console
1
2
3
```
Nested Loops

- Very simple for loop

```java
public class SimpleLoop {
    public static void main (String[] args) {
        for (int i = 1; i <= 3; i++) {
            System.out.println(i);
        }
    }
}
```

- What if for every number below, want multiplication table of value times 2, x3, etc?

```
1 2 3
2 4 6
3 6 9
```

- For every number printed by loop above
  - want another loop to print numbers in a row, instead

```
1 2 3
2 4 6
3 6 9
```

How do we do that?

---

Nested Loops

- Put a loop inside a loop
  - trace to see how it works

```java
public class NestedLoop {
    public static void main (String[] args) {
        for (int i = 1; i <= 3; i++) {
            for (int j = 1; j <= 3; j++) {
                System.out.print((i * j) + " ");
            }
            System.out.println();
        }
    }
}
```

---

Nested Loops

- Put a loop inside a loop
  - trace to see how it works

```java
public class NestedLoop {
    public static void main (String[] args) {
        for (int i = 1; i <= 3; i++) {
            for (int j = 1; j <= 3; j++) {
                System.out.print((i * j) + " ");
            }
            System.out.println();
        }
    }
}
```
Nested Loops

- Put a loop inside a loop
- trace to see how it works

```java
public class NestedLoop {
    public static void main (String[] args) {
        for (int i = 1; i <= 3; i++) {
            for (int j = 1; j <= 3; j++) {
                System.out.print((i * j) + " ");
            }
            System.out.println();
        }
    }
}
```
Nested Loops

- Put a loop inside a loop
  - trace to see how it works

```
public class NestedLoop {
    public static void main(String[] args) {
        for (int i = 1; i <= 3; i++) {
            for (int j = 1; j <= 3; j++) {
                System.out.print((i * j) + " ");
                System.out.println();
            }
        }
    }
}
```

1  2  3

Nested Loops

- Put a loop inside a loop
  - trace to see how it works

```
public class NestedLoop {
    public static void main(String[] args) {
        for (int i = 1; i <= 3; i++) {
            for (int j = 1; j <= 3; j++) {
                System.out.print((i * j) + " ");
                System.out.println();
            }
        }
    }
}
```

1  2  3

Nested Loops

- Put a loop inside a loop
  - trace to see how it works

```
public class NestedLoop {
    public static void main(String[] args) {
        for (int i = 1; i <= 3; i++) {
            for (int j = 1; j <= 3; j++) {
                System.out.print((i * j) + " ");
                System.out.println();
            }
        }
    }
}
```

1  2  3

Nested Loops

- Put a loop inside a loop
  - trace to see how it works

```
public class NestedLoop {
    public static void main(String[] args) {
        for (int i = 1; i <= 3; i++) {
            for (int j = 1; j <= 3; j++) {
                System.out.print((i * j) + " ");
                System.out.println();
            }
        }
    }
}
```

1  2  3
Nested Loops

- Put a loop inside a loop
- trace to see how it works

```java
public class NestedLoop
{
    public static void main(String[] args)
    {
        for (int i = 1; i <= 3; i++)
        {
            for (int j = 1; j <= 3; j++)
            {
                System.out.print((i * j) + " ");
                System.out.println();
            }
        }
    }
}
```
Nested Loops

- Put a loop inside a loop
- trace to see how it works

```java
public class NestedLoop {
    public static void main(String[] args) {
        for (int i = 1; i <= 3; i++) {
            for (int j = 1; j <= 3; j++) {
                System.out.print((i * j) + "  ");
            }
            System.out.println();
        }
    }
}
```

1 2 3
2 3 1
3 2 4

Nested Loops

- Put a loop inside a loop
- trace to see how it works

```java
public class NestedLoop {
    public static void main(String[] args) {
        for (int i = 1; i <= 3; i++) {
            for (int j = 1; j <= 3; j++) {
                System.out.print((i * j) + "  ");
            }
            System.out.println();
        }
    }
}
```

1 2 3
2 3 1
3 2 4

Nested Loops

- Put a loop inside a loop
- trace to see how it works

```java
public class NestedLoop {
    public static void main(String[] args) {
        for (int i = 1; i <= 3; i++) {
            for (int j = 1; j <= 3; j++) {
                System.out.print((i * j) + "  ");
            }
            System.out.println();
        }
    }
}
```

1 2 3
2 3 4
3 1 2

Nested Loops

- Put a loop inside a loop
- trace to see how it works

```java
public class NestedLoop {
    public static void main(String[] args) {
        for (int i = 1; i <= 3; i++) {
            for (int j = 1; j <= 3; j++) {
                System.out.print((i * j) + "  ");
            }
            System.out.println();
        }
    }
}
```

1 2 3
2 3 4
3 1 2

Nested Loops

- Put a loop inside a loop
- trace to see how it works

```java
public class NestedLoop {
    public static void main(String[] args) {
        for (int i = 1; i <= 3; i++) {
            for (int j = 1; j <= 3; j++) {
                System.out.print((i * j) + "  ");
            }
            System.out.println();
        }
    }
}
```

1 2 3
2 3 4
3 1 2

Nested Loops

- Put a loop inside a loop
- trace to see how it works

```java
public class NestedLoop {
    public static void main(String[] args) {
        for (int i = 1; i <= 3; i++) {
            for (int j = 1; j <= 3; j++) {
                System.out.print((i * j) + "  ");
            }
            System.out.println();
        }
    }
}
```

1 2 3
2 3 4
3 1 2
Nested Loops

- Put a loop inside a loop
  - trace to see how it works

```java
public class NestedLoop {
    public static void main (String[] args) {
        for (int i = 1; i <= 3; i++) {
            for (int j = 1; j <= 3; j++) {
                System.out.print((i * j) + " ");
            }
            System.out.println();
        }
    }
}
```

1 2
2 4 6

12 3
24 6

Nested Loops

- Put a loop inside a loop
  - trace to see how it works

```java
public class NestedLoop {
    public static void main (String[] args) {
        for (int i = 1; i <= 3; i++) {
            for (int j = 1; j <= 3; j++) {
                System.out.print((i * j) + " ");
            }
            System.out.println();
        }
    }
}
```

1 2
2 4 6

1 2 3
2 4 6

Nested Loops

- Put a loop inside a loop
  - trace to see how it works

```java
public class NestedLoop {
    public static void main (String[] args) {
        for (int i = 1; i <= 3; i++) {
            for (int j = 1; j <= 3; j++) {
                System.out.print((i * j) + " ");
            }
            System.out.println();
        }
    }
}
```

1 2
2 4 6

1 2 3
2 4 6

Nested Loops

- Put a loop inside a loop
  - trace to see how it works

```java
public class NestedLoop {
    public static void main (String[] args) {
        for (int i = 1; i <= 3; i++) {
            for (int j = 1; j <= 3; j++) {
                System.out.print((i * j) + " ");
            }
            System.out.println();
        }
    }
}
```

1 2
2 4 6

1 2 3
2 4 6

Nested Loops

- Put a loop inside a loop
  - trace to see how it works

```java
public class NestedLoop {
    public static void main (String[] args) {
        for (int i = 1; i <= 3; i++) {
            for (int j = 1; j <= 3; j++) {
                System.out.print((i * j) + " ");
            }
            System.out.println();
        }
    }
}
```

1 2
2 4 6

1 2 3
2 4 6

Nested Loops

- Put a loop inside a loop
  - trace to see how it works

```java
public class NestedLoop {
    public static void main (String[] args) {
        for (int i = 1; i <= 3; i++) {
            for (int j = 1; j <= 3; j++) {
                System.out.print((i * j) + " ");
            }
            System.out.println();
        }
    }
}
```

1 2
2 4 6

1 2 3
2 4 6

Nested Loops

- Put a loop inside a loop
  - trace to see how it works

```java
public class NestedLoop {
    public static void main (String[] args) {
        for (int i = 1; i <= 3; i++) {
            for (int j = 1; j <= 3; j++) {
                System.out.print((i * j) + " ");
            }
            System.out.println();
        }
    }
}
```

1 2
2 4 6

1 2 3
2 4 6

Nested Loops

- Put a loop inside a loop
  - trace to see how it works

```java
public class NestedLoop {
    public static void main (String[] args) {
        for (int i = 1; i <= 3; i++) {
            for (int j = 1; j <= 3; j++) {
                System.out.print((i * j) + " ");
            }
            System.out.println();
        }
    }
}
```

1 2
2 4 6

1 2 3
2 4 6

Nested Loops

- Put a loop inside a loop
  - trace to see how it works

```java
public class NestedLoop {
    public static void main (String[] args) {
        for (int i = 1; i <= 3; i++) {
            for (int j = 1; j <= 3; j++) {
                System.out.print((i * j) + " ");
            }
            System.out.println();
        }
    }
}
```

1 2
2 4 6

1 2 3
2 4 6

Nested Loops

- Put a loop inside a loop
  - trace to see how it works

```java
public class NestedLoop {
    public static void main (String[] args) {
        for (int i = 1; i <= 3; i++) {
            for (int j = 1; j <= 3; j++) {
                System.out.print((i * j) + " ");
            }
            System.out.println();
        }
    }
}
```

1 2
2 4 6

1 2 3
2 4 6

Nested Loops

- Put a loop inside a loop
  - trace to see how it works

```java
public class NestedLoop {
    public static void main (String[] args) {
        for (int i = 1; i <= 3; i++) {
            for (int j = 1; j <= 3; j++) {
                System.out.print((i * j) + " ");
            }
            System.out.println();
        }
    }
}
```

1 2
2 4 6

1 2 3
2 4 6

Nested Loops

- Put a loop inside a loop
  - trace to see how it works

```java
public class NestedLoop {
    public static void main (String[] args) {
        for (int i = 1; i <= 3; i++) {
            for (int j = 1; j <= 3; j++) {
                System.out.print((i * j) + " ");
            }
            System.out.println();
        }
    }
}
```

1 2
2 4 6

1 2 3
2 4 6

Nested Loops

- Put a loop inside a loop
  - trace to see how it works

```java
public class NestedLoop {
    public static void main (String[] args) {
        for (int i = 1; i <= 3; i++) {
            for (int j = 1; j <= 3; j++) {
                System.out.print((i * j) + " ");
            }
            System.out.println();
        }
    }
}
```

1 2
2 4 6

1 2 3
2 4 6

Nested Loops

- Put a loop inside a loop
  - trace to see how it works

```java
public class NestedLoop {
    public static void main (String[] args) {
        for (int i = 1; i <= 3; i++) {
            for (int j = 1; j <= 3; j++) {
                System.out.print((i * j) + " ");
            }
            System.out.println();
        }
    }
}
```

1 2
2 4 6

1 2 3
2 4 6

Nested Loops

- Put a loop inside a loop
  - trace to see how it works

```java
public class NestedLoop {
    public static void main (String[] args) {
        for (int i = 1; i <= 3; i++) {
            for (int j = 1; j <= 3; j++) {
                System.out.print((i * j) + " ");
            }
            System.out.println();
        }
    }
}
```

1 2
2 4 6

1 2 3
2 4 6
Nested Loops

- Put a loop inside a loop
  - trace to see how it works

```java
public class NestedLoop {
    public static void main (String[] args) {
        for (int i = 1; i <= 3; i++) {
            for (int j = 1; j <= 3; j++) {
                System.out.print((i * j) + " ");
            }
            System.out.println();
        }
    }
}
```

1 2 3
2 4 6
3 6 _
Nested Loops

- Put a loop inside a loop
  - trace to see how it works

```java
public class NestedLoop {
    public static void main (String[] args) {
        for (int i = 1; i <= 3; i++) {
            for (int j = 1; j <= 3; j++) {
                System.out.print((i * j) + "  ");
            }
            System.out.println();
        }
    }
}
```

1  2  3
2  4  6
3  6  9

Nested Loops

- Put a loop inside a loop
  - trace to see how it works

```java
public class NestedLoop {
    public static void main (String[] args) {
        for (int i = 1; i <= 3; i++) {
            for (int j = 1; j <= 3; j++) {
                System.out.print((i * j) + "  ");
            }
            System.out.println();
        }
    }
}
```

1  2  3
2  4  6
3  6  9

Nested Loops

- Put a loop inside a loop
  - trace to see how it works

```java
public class NestedLoop {
    public static void main (String[] args) {
        for (int i = 1; i <= 3; i++) {
            for (int j = 1; j <= 3; j++) {
                System.out.print((i * j) + "  ");
            }
            System.out.println();
        }
    }
}
```

1  2  3
2  4  6
3  6  9

Nested Loops

- Put a loop inside a loop
  - trace to see how it works

```java
public class NestedLoop {
    public static void main (String[] args) {
        for (int i = 1; i <= 3; i++) {
            for (int j = 1; j <= 3; j++) {
                System.out.print((i * j) + "  ");
            }
            System.out.println();
        }
    }
}
```

1  2  3
2  4  6
3  6  9

Nested Loops

- Put a loop inside a loop
  - trace to see how it works

```java
public class NestedLoop {
    public static void main (String[] args) {
        for (int i = 1; i <= 3; i++) {
            for (int j = 1; j <= 3; j++) {
                System.out.print((i * j) + "  ");
            }
            System.out.println();
        }
    }
}
```

1  2  3
2  4  6
3  6  9

Nested Loops

- Put a loop inside a loop
  - trace to see how it works

```java
public class NestedLoop {
    public static void main (String[] args) {
        for (int i = 1; i <= 3; i++) {
            for (int j = 1; j <= 3; j++) {
                System.out.print((i * j) + "  ");
            }
            System.out.println();
        }
    }
}
```

1  2  3
2  4  6
3  6  9

Nested Loops

- Put a loop inside a loop
  - trace to see how it works

```java
public class NestedLoop {
    public static void main (String[] args) {
        for (int i = 1; i <= 3; i++) {
            for (int j = 1; j <= 3; j++) {
                System.out.print((i * j) + "  ");
            }
            System.out.println();
        }
    }
}
```

1  2  3
2  4  6
3  6  9
Nested Loops

- Put a loop inside a loop
- Trace to see how it works

public class NestedLoop
{
    public static void main (String[] args)
    {
        for (int i = 1; i <= 3; i++)
        {
            for (int j = 1; j <= 3; j++)
            {
                System.out.print((i * j) + " ");
            }
            System.out.println();
        }
    }
}

Questions?

- Statements within if-else statements can themselves be if-else statements.
- Use curly braces to be clear.

public class NestTest
{
    public static void main (String[] args)
    {
        int x = 1; int y = 3; int z = 2;
        if (x == y)
        {
            if (y == z)
            {
                System.out.println("all three values the same");
            }
            else
            {
                System.out.println("y is not equal to z");
            }
        }
        else
        {
            System.out.println("x is not equal to y");
        }
    }
}

Body Mass Index (BMI) Calculator

- The Body Mass Index (BMI) is a standard medical test to give a quick and approximate indicator of healthy body weight.
- Your BMI is defined as your "weight" in kilograms divided by the square of your height in meters.
- BMI < 19: Underweight, higher risk of disease
- 19 <= BMI <= 25: Normal
- 25 < BMI <= 30: Overweight, higher risk
- 30 < BMI: Obese, much higher risk of disease

- A common solution (others are possible):
  
  if ( bmi < 19 )
  {
      // underweight statements
  }
  else if ( bmi <= 25 )
  {
      // normal statements
  }
  else if ( bmi <= 30 )
  {
      // overweight statements
  }
  else
  {
      // obese statements
  }
else if Syntax

- Easier to read indented like this:
  ```java
  if ( bmi < 19 )
  {
      // underweight statements
  }
  else if ( bmi <= 25 )
  {
      // normal statements
  }
  else if ( bmi <= 30 )
  {
      // overweight statements
  }
  else
  {
      // obese statements
  }
  ```

Dangling else

- What does this code do?
  ```java
  // print warnings only if BMI not normal
  if (bmi >= 19)
  {
      if (bmi > 25) System.out.println("Over");
      else System.out.println("Under");
  }
  ```

Beware the Dangling else

- What does this code do?
  ```java
  // print warnings only if BMI not normal
  if (bmi >= 19)
  {
      if (bmi > 25) System.out.println("Over");
      else System.out.println("Under");
  }
  ```

- else goes with nearest if
  - (but doesn’t look inside curly braces)
  - common mistake
  - whitespace ignored by compiler

Use Curly Braces to Control Nesting

- What does this code do?
  ```java
  // print warnings only if BMI not normal
  if (bmi >= 19) {
      if (bmi > 25) System.out.println("Over");
      else System.out.println("Under");
  }
  ```

- else goes with nearest if
  - but doesn’t look inside curly braces

Commenting Code

- Conventions
  - explain what classes and methods do
  - plus anywhere that you’ve done something non-obvious
  - usually better to say why than what
    - not useful
      ```java
      int wishes = 3; // set wishes to 3
      ```
    - useful
      ```java
      int wishes = 3; // follow fairy tale convention
      ```

Questions?
**javadoc Comments**

- Specific format for method and class header comments
- running javadoc program will automatically generate HTML documentation

**Rules**

- /** to start, first sentence used for method summary
- @param tag for parameter name and explanation
- @return tag for return value explanation
- other tags: @author, @version, etc.
- */ to end

**Running**

```
% javadoc Die.java
% javadoc *.java
```

---

**javadoc Class Comment Example**

```java
/** Die: simulate rolling a die
 * @author: CPSC 111, Section 206, Spring 05-06
 * @version: Jan 31, 2006
 * This is the final Die code. We started on Jan 24,
 * tested and improved in on Jan 26, and did a final
 * cleanup pass on Jan 31.
 */
```

---

**javadoc Method Comment Example**

```java
/**
 * Sets the die shape, thus the range of values it can roll.
 * @param numSides the number of sides of the die
 * @return the number of sides of the die
 */
public void setSides(int numSides) {
    sides = numSides;
}
```

---

**Questions?**

---

**Midterm #1**

- Solutions are posted.
- Protocol for re-mark requests:
  - re-mark requests must be in writing (paper attached to exam) and submitted to instructor
  - re-marking is for mistakes, not for “I should have gotten more points for my wrong answer.”
  - entire exam re-marked

---

**Midterm #1**

- Raw Score Stats:
  - Average: 57.1, Low: 11, High: 100 (Out of 100)
- Scaling Formula: (raw/100)0.7 * 100

<table>
<thead>
<tr>
<th>Raw Score</th>
<th>Scaled Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>27</td>
</tr>
<tr>
<td>30</td>
<td>43</td>
</tr>
<tr>
<td>45</td>
<td>57</td>
</tr>
<tr>
<td>60</td>
<td>70</td>
</tr>
<tr>
<td>75</td>
<td>82</td>
</tr>
<tr>
<td>90</td>
<td>93</td>
</tr>
</tbody>
</table>