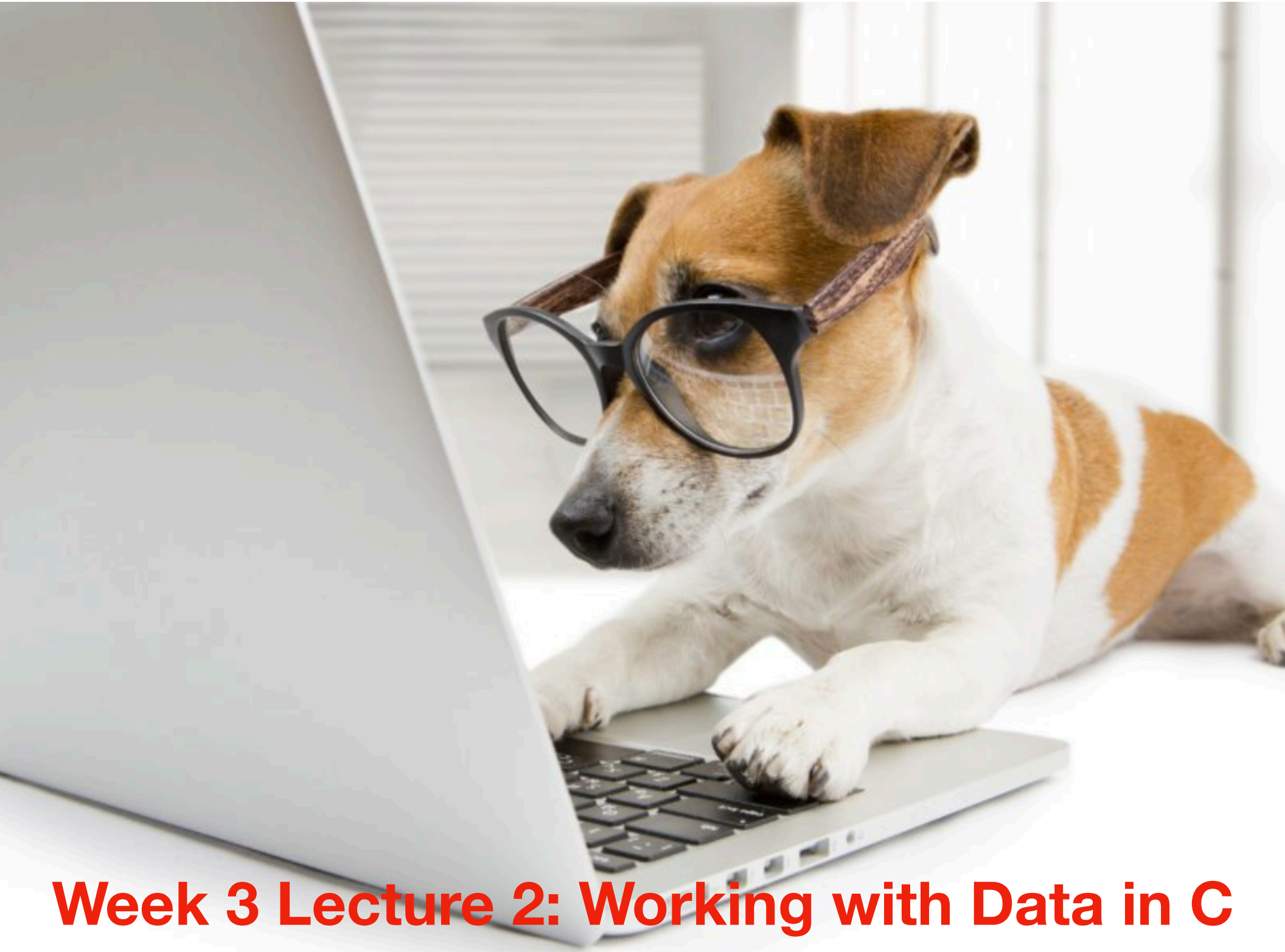


Week 3 Lecture 2: Working with Data in C

Review and Preview

- Types in C
- Storing data in C

- Arithmetic, Assignment, Relational and Logical Operators
- Type casting, sizeof
- The Midterm!

K&R: Chapters 1-3
Sections 1.1 to 1.6
Sections 2.1 to 2.6
Sections 3.1 to 3.6

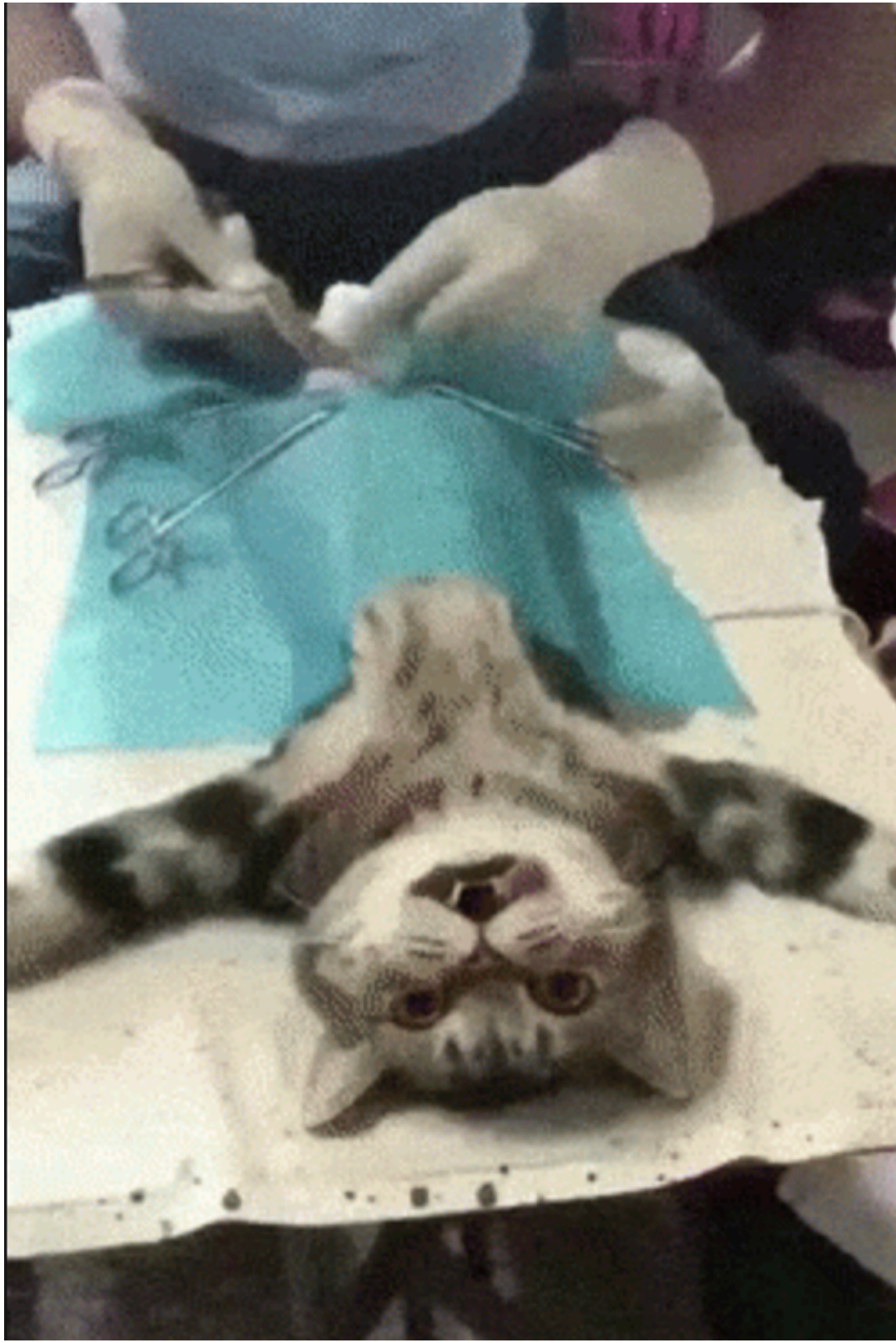
UNIX Command of the Day: `diff`

- `diff` compares files line by line
- Interesting argument flags include:
  - -i ignores case differences in the files.
  - -E ignores changes due to tab expansion.
  - -b ignores changes in the amount of white space.
- If there are differences it outputs them and their location in the two files.

```
Deborahs-MacBook-Pro:Assignment1 debstacey$ diff daysCalculatorA.c daysCalculatorB.c
2a3
> #include <string.h>
5c6
<  *   Program name: daysCalculatorA.c
---
>  *   Program name: daysCalculatorB.c
8,11c9,14
<  *   Function: calculates and outputs the number of days between two dates.
<  *             The output is the number of days.
<  *   Compilation: gcc -ansi -o dayCalculatorA dayCalculatorA.c
<  *   Execution: ./daysCalculatorA 23 8 2019 25 11 2019
---
>  *   Function: Calculates the number of days between two dates.
>  *             It outputs the number of days.  If include is the
>  *             last parameter on the command line it adds the end
>  *             day to the count.
>  *   Compilation: gcc -ansi -o daysCalculatorB daysCalculatorB.c
>  *   Execution: ./daysCalculatorB 23 8 2019 10 10 2019 include
32a36
>    int incFlag = 0;
34,35c38,39
<    if ( argc != 7 ) {
<       printf ( "Usage: ./dates dd1 mm1 yyyy1 dd2 mm2 yyyy2\n" );
---
>    if ( argc < 7 ) {
>       printf ( "Usage: ./dates dd1 mm1 yyyy1 dd2 mm2 yyyy2 <include>\n" );
43a48,52
>       if ( argc == 8 ) {
>          if ( strcmp("include",argv[7]) == 0 ) {
>             incFlag = 1;
>          }
>       }
84a94,96
>    if ( incFlag == 1 ) {
>       numDays++;
>    }
Deborahs-MacBook-Pro:Assignment1 debstacey$
```

Operators in C

- Arithmetic, Assignment, Relational and Logical Operators
- Type casting, sizeof



Arithmetic Operators

- Addition **+**
- Subtraction **–**
- Multiplication *****
- Division **/**
- Remainder after division (modulo division) **%**

Arithmetic Operators: Add, Subtract, Multiply

```
int num1, num2, sum, delta, prod;

num1 = 23;

num2 = 12;

sum = num1 + num2;

delta = num1 - num2;

prod = num1 * num2;
```

```
$ ./arithmetic
23 12 35 11 276
$
```

Increment and Decrement Operators

- There are two operators that change the value of a variable by 1.
  - increment **++**
  - decrement **--**
- They are **unary** operators. They only operate on a single operand (variable).

Increment and Decrement Operators

- Both the increment and decrement operators can be used in prefix and postfix.
- **Prefix**
  - `++x` performs the adding of the one before any other action is taken.

```
int x = 0;

printf ( “++x = %d\n”, ++x );
```

Increment and Decrement Operators

- **Postfix**
  - `x++` performs the adding of the one after any other action is taken.

```
int x = 0;

printf ( “x++ = %d\n”, x++ );
```

```
#include <stdio.h>
int main ( int argc, char *argv[] ) {
    int x;

    x = 0;
    printf ( "x = %d\n", x );
    printf ( "Prefix: ++x = %d\n", ++x );
    printf ( "x = %d\n", x );

    printf ( "----------\n" );

    x = 0;
    printf ( "x = %d\n", x );
    printf ( "Postfix: x++ = %d\n", x++ );
    printf ( "x = %d\n", x );
    return ( 0 );
}
```

```
x = 0
Prefix: ++x = 1
x = 1
---------
x = 0
Postfix: x++ = 0
x = 1
```

Arithmetic Operations: Division and Modulo

- Division produces results based on the types of the operands.
- If both operands are integers then the result is an integer, even if the result has a fractional part - you only see the integer part of the result.
- If either one of the operands are floating point numbers then the result can be a float and thus the fractional part of the division is revealed.

```
#include <stdio.h>
int main ( int argc, char *argv[] ) {
    int n1, n2, n3;
    float n4, n5;
    n1 = 14;
    n2 = 3;
    n4 = 3.0;

    n3 = n1 / n2;   Integer division
    n5 = n1 / n4;   Real or floating point division

    printf ("Integer division: %d / %d = %d\n",
n1, n2, n3 );
    printf ("Real division: %d / %f = %f\n",
n1, n4, n5 );
    return ( 0 );
}
```

Integer division: 14 / 3 = 4
Real division: 14 / 3.000000 = 4.666667

Modulo Division

- The modulo operator `%` computes the remainder.
- The `%` operator can only be used with integers.

```
int n1, n2, n3;
n1 = 14;
n2 = 3;
n3 = n1 % n2;
printf ( "%d %% %d = %d\n", n1, n2, n3 );
printf ( "%d / %d = %d\n", n1, n2, n1/n2 );
```

14 % 3 = 2
14 / 3 = 4

Assignment Operators

- An assignment operator assigns a value to an integer.

```
num1 = 17;
```

- But there are assignment operators that also perform another operation when they are doing the assignment.

`a += b` *same as* `a = a+b` `a -= b` *same as* `a = a-b`

`a *= b` *same as* `a = a*b` `a /= b` *same as* `a = a/b`

`a %= b` *same as* `a = a%b`

Relational Operators

- Relational operators are used to form boolean expressions.
- They are used for decision making in branches and loops.

Equal to `==`	Not equal to `!=`
Greater than `>`	Greater than or equal to `>=`
Less than `<`	Less than or equal to `<=`

Logical Operators

- Logical operators are also used in boolean expressions and form the basis of decision making.
- `&&` Logical AND - both operands must be true for the expression to be true.
- `||` Logical OR - only one operand must be true for the expression to be true.
- `!` Logical NOT - if the operand is false then the expression is true.

Sizeof Operator

- The `sizeof()` operator returns the size of data (data types, variables, array, structures, etc) as an `unsigned long int`.

```
int numArray[12];
float realNumbers[50];
char string[100];

printf ( "sizeof numArray = 12 * %lu = %lu\n",
sizeof(int), sizeof(numArray) );

printf ( "sizeof realNumbers = 50 * %lu = %lu\n",
sizeof(float), sizeof(realNumbers) );

printf ( "sizeof string = 100 * %lu = %lu\n",
sizeof(char), sizeof(string) );
```

Type Casting

- Type casting is the process of converting one type of data to another type.
- Some conversions are done implicitly by the compiler but you can do type casting explicitly and this is programming best practice.
- Format of type casting: (type) expression

```
int num = 12;
float fnum = 2.3;
fnum = (float) num;
num = (int) fnum;
```

```
int num = 12;
int num1;
float fnum = 2.3;
float fnum1;

fnum1 = (float) num;
printf ("num = %d fnum1 = %f\n", num, fnum1);

num1 = (int) fnum;
printf ("fnum = %f num1 = %d\n", fnum, num1);
```

```
num = 12 fnum1 = 12.000000
fnum = 2.300000 num1 = 2
```

The Midterm!

What is on the midterm?

Question Types

Midterm Preview

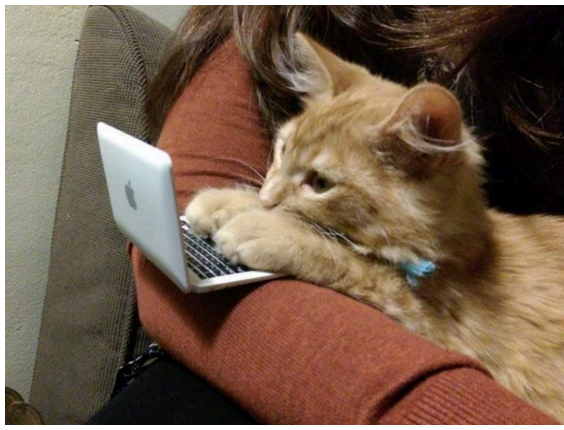
- **When**: Thursday, October 10 during class time (10:15am to 11:15am *or* 4:15pm to 5:15pm)
- **Where**: Roz 103 (lecture room)
- **Length**: 60 minutes
- **Type**: Closed book

Midterm Preview: Types of Content

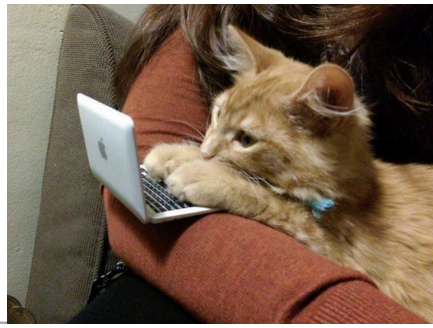
- **General knowledge**
  - definitions, UNIX commands
- **Reading code**
  - Given a command line, what are the values of `argc`, `argv[x]`
  - Given a code snippet, what does it do
- **Writing code**

Example General Knowledge Questions

- UNIX command that returns the current working directory name: ?
- What is an operating system?
- What does VM stand for?
- What does the UNIX command `grep` do?



Example Reading Code Questions



./pgm 12 3 2019 Gregorian

`argc = ?  argv[0] = ?  argv[4] = ?  argv[5] = ?`

What does the following output?

```
for (  i=0; i<2; i++ ) {
   printf ( "%d. %d\n", i+1, i );
}
```

Example Writing Code Question

- Write a `while` loop that prints out the numbers from 1 to 10.
- Write a `for` loop that prints out the numbers 2, 4, 6, 8.
- Write the code that checks to see if a year represented by an integer is a leap year.
- Write code that checks to see if there are 4 command line arguments **after** the program name. If the number of arguments is incorrect print out an appropriate message and then terminate the program.