More choice for Mr C

ESC101: Fundamentals of Computing

Purushottam Kar

Announcements

- Institute holiday on August 22, 2018, Wednesday
  - No lecture, no lab on August 22
  - No extra lecture this week
- Extra lab for Wednesday batches B10, B11, B12, B14
  - Saturday, August 25, 2018, 2PM New Core Labs CC-01, CC-02
- Refer to course schedule calendar on website
  web.cse.iitk.ac.in/users/purushot/courses/esc/2018-19-a/material/schedule.pdf

Announcements

- Extra session for students facing trouble with English lectures but who are comfortable with Hindi
  - Saturday, August 25, 2018, 5PM-6:30PM, New Core Labs CC-02
  - Extra session to be held just after extra lab is over for B10, B11, B12, B14
- Students familiar with other Indian languages, please refer to document on website for names of admins
  web.cse.iitk.ac.in/users/purushot/courses/esc/2018-19-a/material/language.pdf

Announcements

- Marks for week 2 lab released last weekend
- See grade-card for marks
- Go to codebook if you feel regrading is required – option will be there to make regrading request
- See guidelines in last lecture for how autograding done
- Frivolous/useless regrading requests will be penalized
- Minor quiz marks for week 2 and 3 released this week
- Sorry for the delay

The Goldilocks Challenge

The Goldilocks Challenge

Write a program to take a temperature and print

The Goldilocks Challenge

Write a program to take a temperature and print

- Too Cold if temperature is below 22

The Goldilocks Challenge

Write a program to take a temperature and print

- Too Cold if temperature is below 22
- Just Right if between 22 and 27

The Goldilocks Challenge

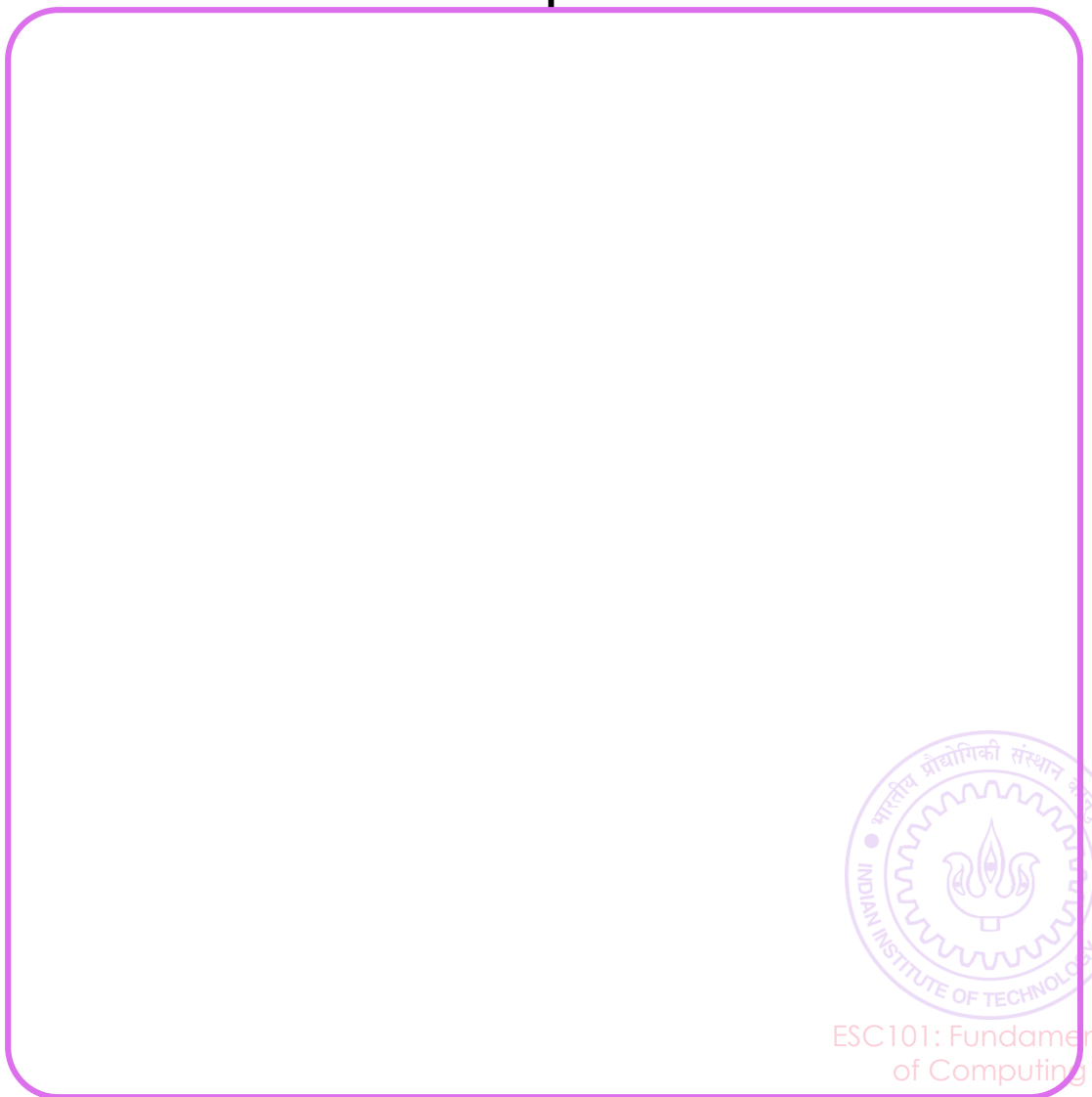
Write a program to take a temperature and print

- Too Cold if temperature is below 22
- Just Right if between 22 and 27
- Too Hot if temperature is above 27

The Goldilocks Challenge

Write a program to take a temperature and print

- Too Cold if temperature is below 22
- Just Right if between 22 and 27
- Too Hot if temperature is above 27



The Goldilocks Challenge

Write a program to take a temperature and print

- Too Cold if temperature is below 22
- Just Right if between 22 and 27
- Too Hot if temperature is above 27

```
if(temp < 22){
```

The Goldilocks Challenge

Write a program to take a temperature and print

- Too Cold if temperature is below 22
- Just Right if between 22 and 27
- Too Hot if temperature is above 27

```
if(temp < 22){


}else{
```

The Goldilocks Challenge

Write a program to take a temperature and print

- Too Cold if temperature is below 22
- Just Right if between 22 and 27
- Too Hot if temperature is above 27

```
if(temp < 22){


}else{









}
```

The Goldilocks Challenge

Write a program to take a temperature and print

- Too Cold if temperature is below 22
- Just Right if between 22 and 27
- Too Hot if temperature is above 27

```
if(temp < 22){
    printf("Too Cold");
}else{




}
```

The Goldilocks Challenge

Write a program to take a temperature and print

- Too Cold if temperature is below 22
- Just Right if between 22 and 27
- Too Hot if temperature is above 27

```
if(temp < 22){
    printf("Too Cold");
}else{
    if (22 <= temp <= 27){




}
```

The Goldilocks Challenge

Write a program to take a temperature and print

- Too Cold if temperature is below 22
- Just Right if between 22 and 27
- Too Hot if temperature is above 27

```
if(temp < 22){
    printf("Too Cold");
}else{
    if (22 <= temp <= 27){
        printf("Just Right");




}
```

The Goldilocks Challenge

Write a program to take a temperature and print

- Too Cold if temperature is below 22
- Just Right if between 22 and 27
- Too Hot if temperature is above 27

```
if(temp < 22){
    printf("Too Cold");
}else{
    if (22 <= temp <= 27){
        printf("Just Right");
    }else{



}
```

The Goldilocks Challenge

Write a program to take a temperature and print

- Too Cold if temperature is below 22
- Just Right if between 22 and 27
- Too Hot if temperature is above 27

```
if(temp < 22){
    printf("Too Cold");
}else{
    if (22 <= temp <= 27){
        printf("Just Right");
    }else{
        printf("Too Hot");


}
```

The Goldilocks Challenge

Write a program to take a temperature and print

- Too Cold if temperature is below 22
- Just Right if between 22 and 27
- Too Hot if temperature is above 27

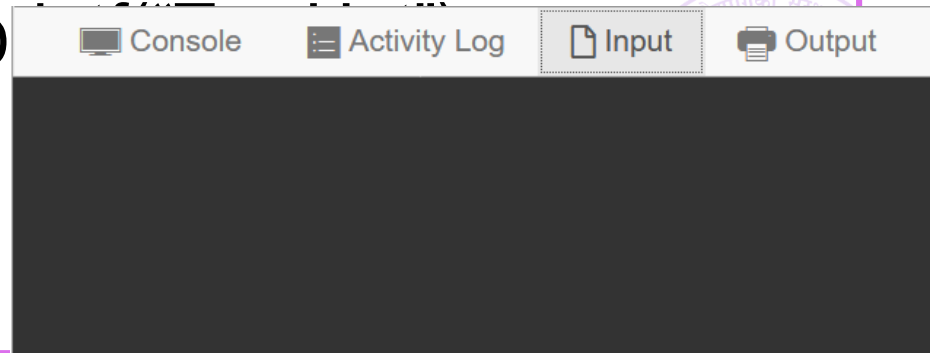
```
if(temp < 22){
    printf("Too Cold");
}else{
    if (22 <= temp <= 27){
        printf("Just Right");
    }else{
        printf("Too Hot");
    }
}
```

The Goldilocks Challenge

Write a program to take a temperature and print

- Too Cold if temperature is below 22
- Just Right if between 22 and 27
- Too Hot if temperature is above 27

```
if(temp < 22){
    printf("Too Cold");
}else{
    if (22 <= temp <= 27){
        printf("Just Right");
    }else{
        p
    }
}
```

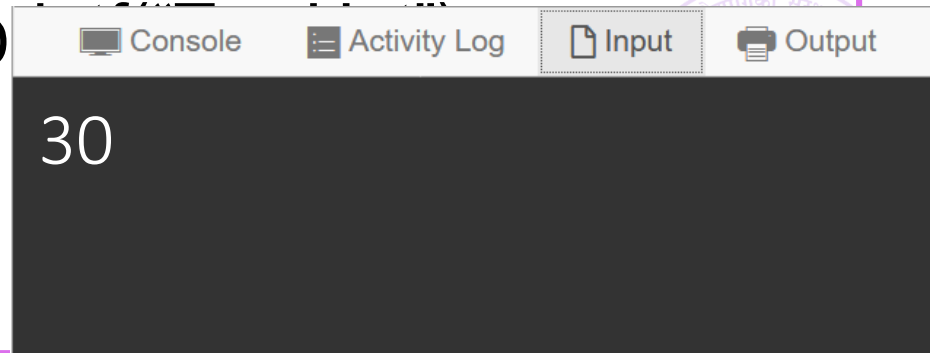


The Goldilocks Challenge

Write a program to take a temperature and print

- Too Cold if temperature is below 22
- Just Right if between 22 and 27
- Too Hot if temperature is above 27

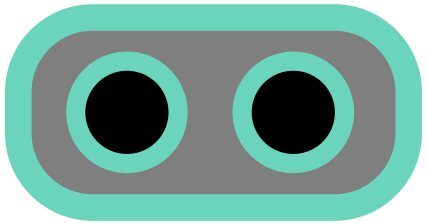
```
if(temp < 22){
    printf("Too Cold");
}else{
    if (22 <= temp <= 27){
        printf("Just Right");
    }else{
        p
    }
}
```



The Goldilocks Challenge

Write a program to take a temperature and print

- Too Cold if temperature is below 22
- Just Right if between 22 and 27
- Too Hot if temperature is above 27



```
if(temp < 22){
    printf("Too Cold");
}else{
    if (22 <= temp <= 27){
        printf("Just Right");
    }else{
        p
    }
}
```

Console | Activity Log | Input | Output

30

The Goldilocks Challenge

Write a program to take a temperature and print

- Too Cold if temperature is below 22
- Just Right if between 22 and 27
- Too Hot if temperature is above 27


```
if(temp < 22){
    printf("Too Cold");
}else{
    if (22 <= temp <= 27){
        printf("Just Right");
    }else{
        p
    }
}
```

Console | Activity Log | Input | Output

30

The Goldilocks Challenge

Write a program to take a temperature and print

- Too Cold if temperature is below 22
- Just Right if between 22 and 27
- Too Hot if temperature is above 27

```
if(temp < 22){
    printf("Too Cold");
}else{
    if (22 <= temp <= 27){
        printf("Just Right");
    }else{
        p
    }
}
```

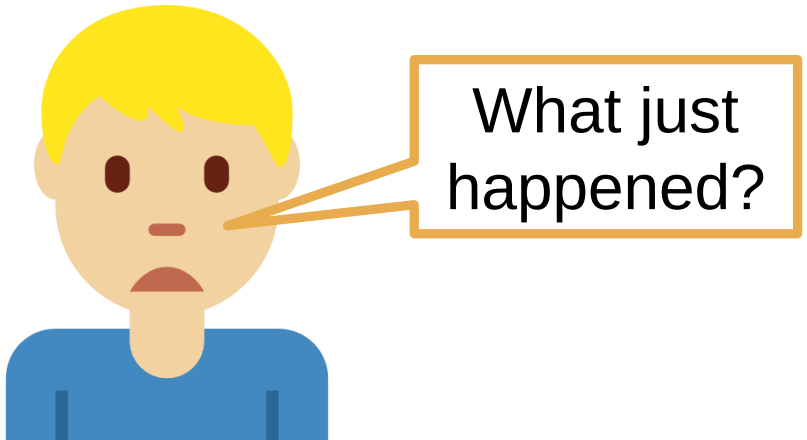
Console | Activity Log | Input | Output

30

The Goldilocks Challenge

Write a program to take a temperature and print

- Too Cold if temperature is below 22
- Just Right if between 22 and 27
- Too Hot if temperature is above 27




```
if(temp < 22){
    printf("Too Cold");
}else{
    if (22 <= temp <= 27){
        printf("Just Right");
    }else{
        p
    }
}
```

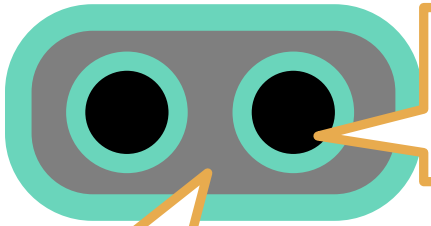
Console | Activity Log | Input | Output

30

The Goldilocks Challenge

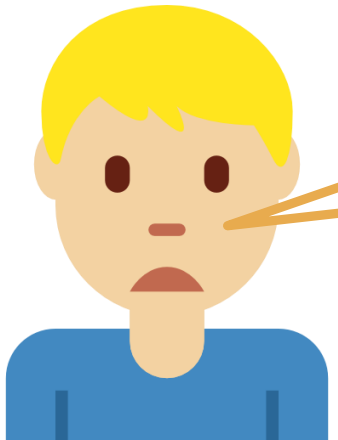
Write a program to take a temperature and print

Too Cold if temperature is below 22
Just Right if between 22 and 27
Too Hot if temperature is above 27



Remember, expressions generate values

Just Right



What just happened?

```
if(temp < 22){
    printf("Too Cold");
}else{
    if (22 <= temp <= 27){
        printf("Just Right");
    }else{
        p
    }
}
```

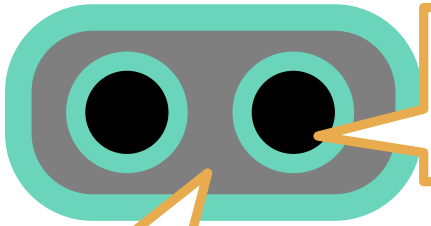
Console | Activity Log | Input | Output

30

The Goldilocks Challenge

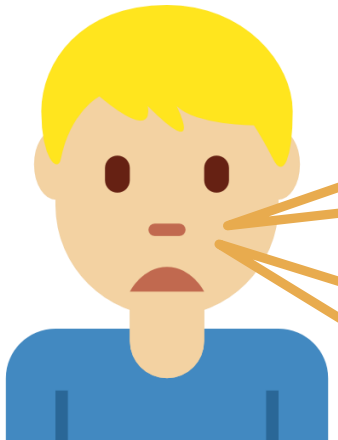
Write a program to take a temperature and print

Too Cold if temperature is below 22
Just Right if between 22 and 27
Too Hot if temperature is above 27



Remember, expressions generate values

Just Right



What just happened?

What value does temp < 22 generate?

```
if(temp < 22){
    printf("Too Cold");
}else{
    if (22 <= temp <= 27){
        printf("Just Right");
    }else{
        p
    }
}
```

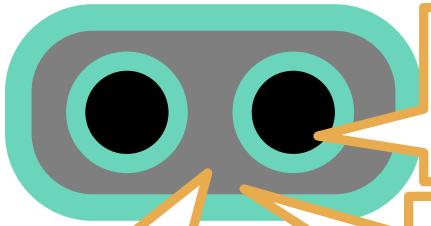
Console | Activity Log | Input | Output

30

The Goldilocks Challenge

Write a program to take a temperature and print

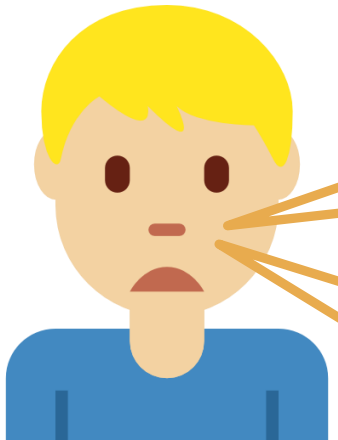
Too Cold if temperature is below 22
Just Right if between 22 and 27
Too Hot if temperature is above 27



Remember, expressions generate values

Just Right

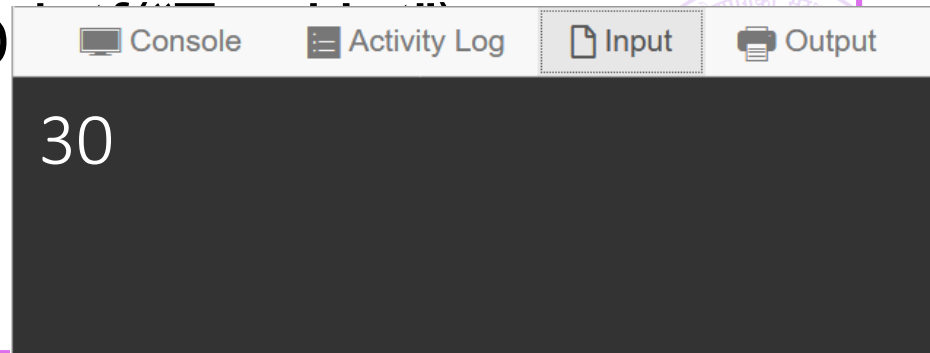
If temp is less than 22, it generates value 1. If temp is greater than or equal to 22, it generates value 0



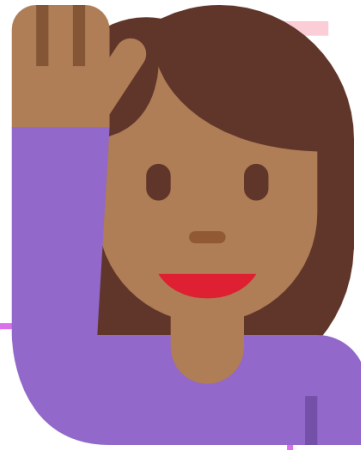
What just happened?

What value does temp < 22 generate?

```
if(temp < 22){
    printf("Too Cold");
}else{
    if (22 <= temp <= 27){
        printf("Just Right");
    }else{
        p
    }
}
```

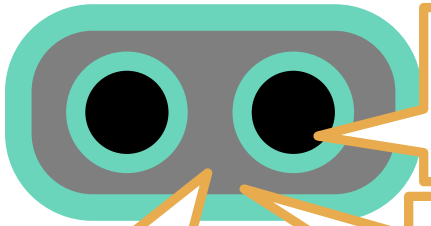


The Goldilocks Challenge



Write a program to take a temperature and print

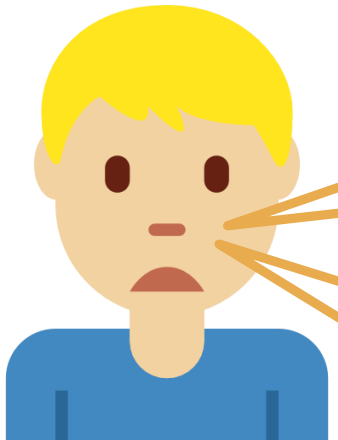
- Too Cold if temperature is below 22
- Just Right if between 22 and 27
- Too Hot if temperature is above 27



Remember, expressions generate values

Just Right

If temp is less than 22, it generates value 1. If temp is greater than or equal to 22, it generates value 0



What just happened?

What value does temp < 22 generate?

```
if(temp < 22){
    printf("Too Cold");
}else{
    if (22 <= temp <= 27){
        printf("Just Right");
    }else{
        p
    }
}
```

Console | Activity Log | Input | Output

30

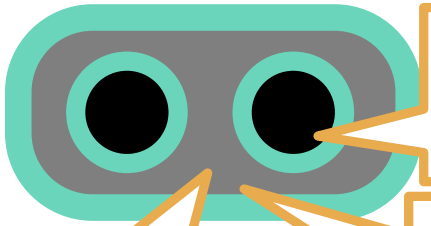
The Goldilocks Challenge

Write a program to take

Too Cold if temperature is bel

Just Right if between 22 and 27

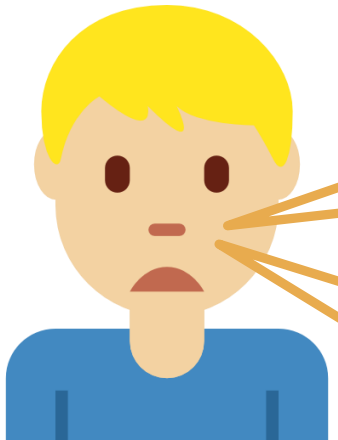
Too Hot if temperature is above 27



Remember, expressions generate values

Just Right

If temp is less than 22, it generates value 1. If temp is greater than or equal to 22, it generates value 0

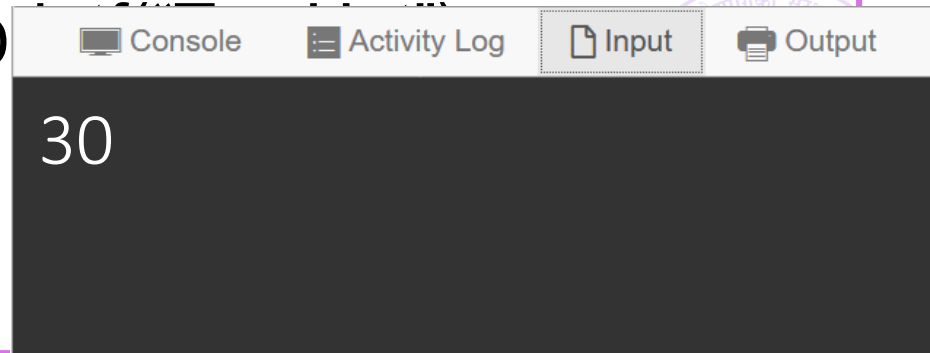


What just happened?

What value does temp < 22 generate?

Mr C considers 0 to be FALSE and 1 (or anything non-zero) to be TRUE

```
    printf("Too Cold");
}else{
    if (22 <= temp <= 27){
        printf("Just Right");
    }else{
        p
    }
}
```



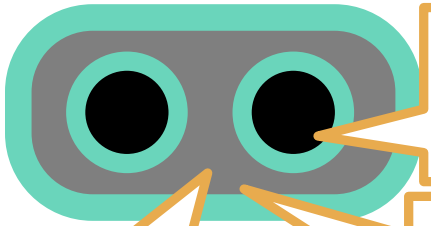
The Goldilocks Challenge

Write a program to take

Too Cold if temperature is bel

Just Right if between 22 and 27

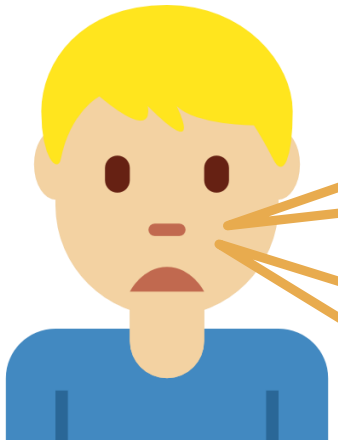
Too Hot if temperature is above 27



Remember, expressions generate values

Just Right

If temp is less than 22, it generates value 1. If temp is greater than or equal to 22, it generates value 0



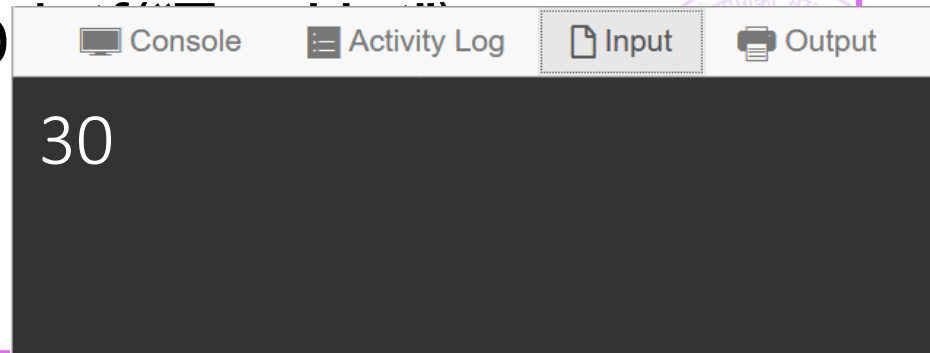
What just happened?

What value does temp < 22 generate?

Mr C considers 0 to be FALSE and 1 (or anything non-zero) to be TRUE

If the expression inside if(…) evaluates to 1 or something non-zero, Mr C executes the if part. If the expression evaluates to 0, Mr C executes the else part

```
if (22 < temp < 27){
    printf("Just Right");
}else{
    p
    }
}
```



Complex Relational Expressions

Complex Relational Expressions

<, <=, ==, >, >=, != are called *relational operators*

Complex Relational Expressions

<, <=, ==, >, >=, != are called *relational operators*

Expressions containing these operators generate 0 or 1

Complex Relational Expressions

<, <=, ==, >, >=, != are called *relational operators*

Expressions containing these operators generate 0 or 1

All have left to right associativity (just like +, -, *, /)

Complex Relational Expressions

<, <=, ==, >, >=, != are called *relational operators*

Expressions containing these operators generate 0 or 1

All have left to right associativity (just like +, -, *, /)

22 <= temp <= 27 became ((22 <= temp) <= 27)

Complex Relational Expressions

<, <=, ==, >, >=, != are called *relational operators*

Expressions containing these operators generate 0 or 1

All have left to right associativity (just like +, -, *, /)

22 <= temp <= 27 became ((22 <= temp) <= 27)

When we entered 30, Mr C evaluated ((22 <= 30) <= 27)

Complex Relational Expressions

<, <=, ==, >, >=, != are called *relational operators*

Expressions containing these operators generate 0 or 1

All have left to right associativity (just like +, -, *, /)

22 <= temp <= 27 became ((22 <= temp) <= 27)

When we entered 30, Mr C evaluated ((22 <= 30) <= 27)

This became (1 <= 27) which is true so the final result is 1

Complex Relational Expressions

<, <=, ==, >, >=, != are called *relational operators*

Expressions containing these operators generate 0 or 1

All have left to right associativity (just like +, -, *, /)

22 <= temp <= 27 became ((22 <= temp) <= 27)

When we entered 30, Mr C evaluated ((22 <= 30) <= 27)

This became (1 <= 27) which is true so the final result is 1

This is why Mr C printed Just Right even when temp = 30

Complex Relational Expressions

<, <=, ==, >, >=, != are called *relational operators*

Expressions containing these operators generate 0 or 1

All have left to right associativity (just like +, -, *, /)

22 <= temp <= 27 became ((22 <= temp) <= 27)

When we entered 30, Mr C evaluated ((22 <= 30) <= 27)

This became (1 <= 27) which is true so the final result is 1

This is why Mr C printed Just Right even when temp = 30

Complex Relational Expressions

<, <=, ==, >, >=, != are called *relational operators*

Expressions containing these operators generate 0 or 1

All have left to right associativity (just like +, -, *, /)

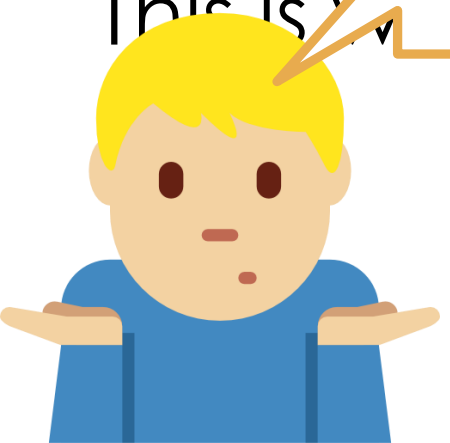
22 <= temp <= 27 became ((22 <= temp) <= 27)

When we entered 30, Mr C evaluated ((22 <= 30) <= 27)

This bec is true so the final result is 1

This is Right even when temp = 30

So how do I get Mr C to do something when temperature is between 22 and 27?



Logical Operators

Logical Operators

Used to create powerful conditions and choices

Logical Operators

Used to create powerful conditions and choices

If we want temp >= 22 AND temp <= 27

Logical Operators

Used to create powerful conditions and choices

If we want temp >= 22 AND temp <= 27

```
if((22 <= temp) && (temp <= 27)){ ... }
```

Logical Operators

Used to create powerful conditions and choices

If we want temp >= 22 AND temp <= 27

```
if((22 <= temp) && (temp <= 27)){ ... }
if((temp >= 22) && (temp <= 27)){ ... }
```

Logical Operators

Used to create powerful conditions and choices

If we want temp >= 22 AND temp <= 27

```
if((22 <= temp) && (temp <= 27)){ ... }
if((temp >= 22) && (temp <= 27)){ ... }
```

If we want temp >= 22 OR temp <= 27

Logical Operators

Used to create powerful conditions and choices

If we want temp >= 22 AND temp <= 27

```
if((22 <= temp) && (temp <= 27)){ ... }
if((temp >= 22) && (temp <= 27)){ ... }
```

If we want temp >= 22 OR temp <= 27

```
if((temp >= 22) || (temp <= 27)){ ... }
```

Logical Operators

Used to create powerful conditions and choices

If we want temp >= 22 AND temp <= 27

```
if((22 <= temp) && (temp <= 27)){ ... }
if((temp >= 22) && (temp <= 27)){ ... }
```

If we want temp >= 22 OR temp <= 27

```
if((temp >= 22) || (temp <= 27)){ ... }
```

If we want NOT a % 2 == 0 (to select odd numbers)

Logical Operators

Used to create powerful conditions and choices

If we want temp >= 22 AND temp <= 27

```
if((22 <= temp) && (temp <= 27)){ ... }
if((temp >= 22) && (temp <= 27)){ ... }
```

If we want temp >= 22 OR temp <= 27

```
if((temp >= 22) || (temp <= 27)){ ... }
```

If we want NOT a % 2 == 0 (to select odd numbers)

```
if(!(a %2 == 0) ){ ... }
```

Logical Operators

Used to create powerful conditions and choices

If we want temp >= 22 AND temp <= 27

```
if((22 <= temp) && (temp <= 27)){ ... }
if((temp >= 22) && (temp <= 27)){ ... }
```

If we want temp >= 22 OR temp <= 27

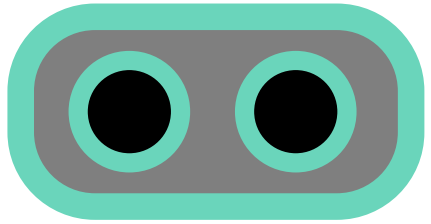
```
if((temp >= 22) || (temp <= 27)){ ... }
```

If we want NOT a % 2 == 0 (to select odd numbers)

```
if(!(a %2 == 0) ){ ... }
if(a %2 != 0){ ... }
```

Logical Operators

Used to create powerful conditions and choices



If we want temp >= 22 AND temp <= 27

```
if((22 <= temp) && (temp <= 27)){ ... }
if((temp >= 22) && (temp <= 27)){ ... }
```

If we want temp >= 22 OR temp <= 27

```
if((temp >= 22) || (temp <= 27)){ ... }
```

If we want NOT a % 2 == 0 (to select odd numbers)

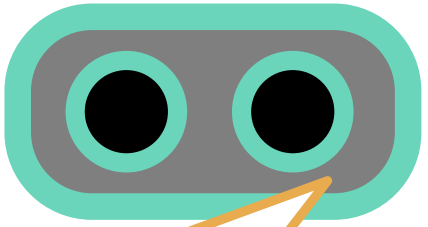
```
if(!(a %2 == 0) ){ ... }
if(a %2 != 0){ ... }
```

Logical Operators

Used to create powerful conditions and choices

If we want temp >= 22 AND temp <= 27

```
if((22 <= temp) && (temp <= 27)){ ... }
if((temp >= 22) && (temp <= 27)){ ... }
```



Bracket yourself to avoid errors and confusion

If we want temp >= 22 OR temp <= 27

```
if((temp >= 22) || (temp <= 27)){ ... }
```

If we want NOT a % 2 == 0 (to select odd numbers)

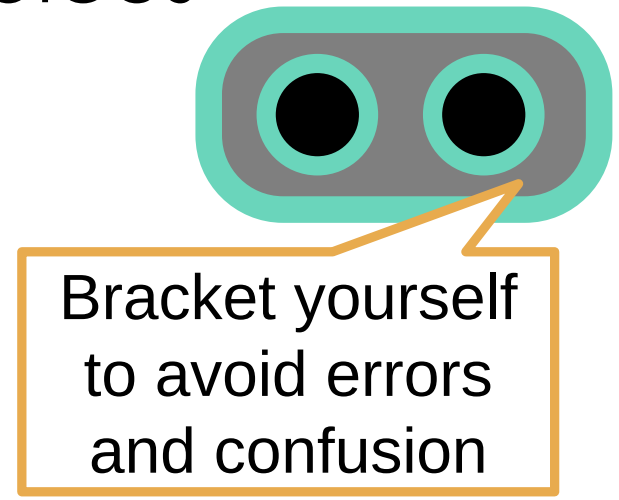
```
if(!(a %2 == 0) ){ ... }
if(a %2 != 0){ ... }
```

Logical Operators

Used to create powerful conditions and choices

If we want temp >= 22 AND temp <= 27

```
if((22 <= temp) && (temp <= 27)){ ... }
if((temp >= 22) && (temp <= 27)){ ... }
```

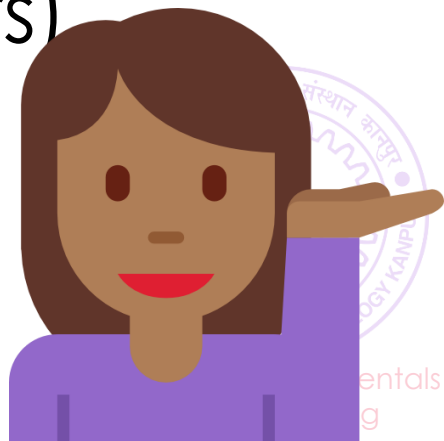


If we want temp >= 22 OR temp <= 27

```
if((temp >= 22) || (temp <= 27)){ ... }
```

If we want NOT a % 2 == 0 (to select odd numbers)

```
if(!(a %2 == 0) ){ ... }
if(a %2 != 0){ ... }
```

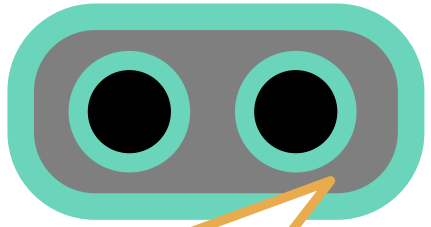


Logical Operators

Used to create powerful conditions and choices

If we want temp >= 22 AND temp <= 27

```
if((22 <= temp) && (temp <= 27)){ ... }
if((temp >= 22) && (temp <= 27)){ ... }
```



Bracket yourself to avoid errors and confusion

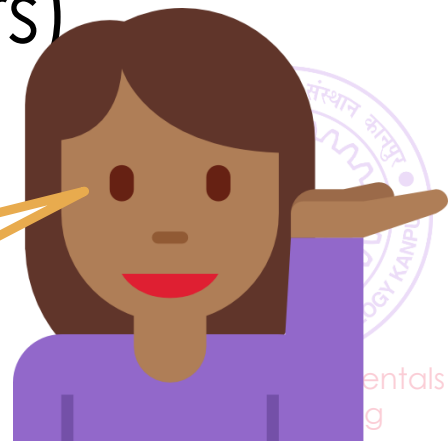
If we want temp >= 22 OR temp <= 27

```
if((temp >= 22) || (temp <= 27)){ ... }
```

If we want NOT a % 2 == 0 (to select odd numbers)

```
if(!(a %2 == 0) ){ ... }
if(a %2 != 0){ ... }
```

If you don't put brackets, Mr C will put brackets according to his table



Logical Operators

Used to create powerful conditions and choices

If we want temp >= 22 AND temp <= 27

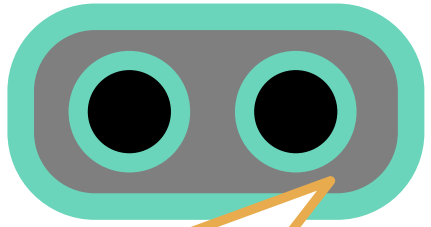
```
if((22 <= temp) && (temp <= 27)){ ... }
if((temp >= 22) && (temp <= 27)){ ... }
```

If we want temp >= 22 OR temp <= 27

```
if((temp >= 22) || (temp <= 27)){ ... }
```

If we want NOT a % 2 == 0 (to select odd numbers)

```
if(!(a %2 == 0) ){ ... }
if(a %2 != 0){ ... }
```



Bracket yourself to avoid errors and confusion

Lets show them the new tables

If you don't put brackets, Mr C will put brackets according to his table

Logical Operators

Used to create powerful conditions and choices

If we want temp >= 22 AND temp <= 27

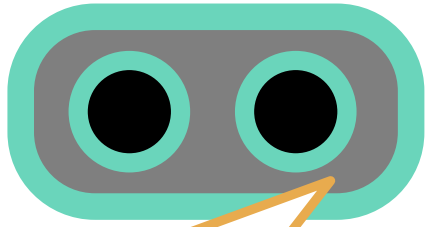
```
if((22 <= temp) && (temp <= 27)){ ... }
if((temp >= 22) && (temp <= 27)){ ... }
```

If we want temp >= 22 OR temp <= 27

```
if((temp >= 22) || (temp <= 27)){ ... }
```

If we want NOT a % 2 == 0 (to select odd numbers)

```
if(!(a %2 == 0) ){ ... }
if(a %2 != 0){ ... }
```

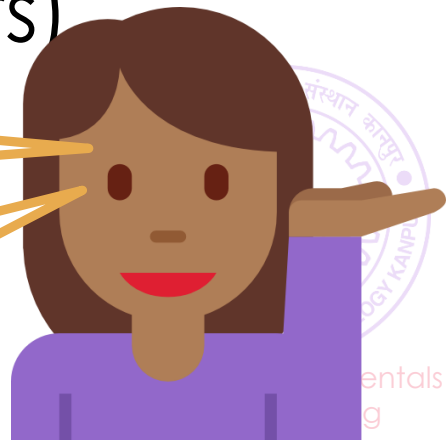


Bracket yourself to avoid errors and confusion

Lets show them the new tables

Yes, but first a bit more on &&, || and !

If you don't put brackets, Mr C will put brackets according to his table



Logical Operators

Used to create powerful conditions and

If we want temp >= 22 AND temp <= 27

```
if((22 <= temp) && (temp <= 27)){ ... }
if((temp >= 22) && (temp <= 27)){ ... }
```

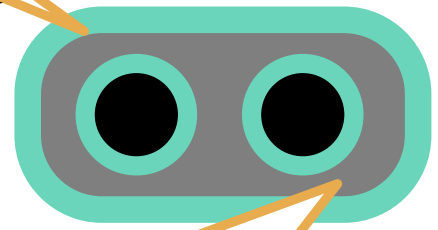
If we want temp >= 22 OR temp <= 27

```
if((temp >= 22) || (temp <= 27)){ ... }
```

If we want NOT a % 2 == 0 (to select odd numbers)

```
if(!(a %2 == 0) ){ ... }
if(a %2 != 0){ ... }
```

Good idea -

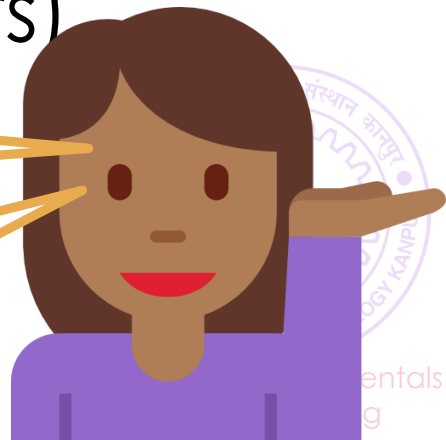


Bracket yourself to avoid errors and confusion

Lets show them the new tables

Yes, but first a bit more on &&, || and !

If you don't put brackets, Mr C will put brackets according to his table



Values generated by Logical Ops

Values generated by Logical Ops

a = 1 if good marks, else 0, b = 1 if good attendance, else 0

Values generated by Logical Ops

a = 1 if good marks, else 0, b = 1 if good attendance, else 0

Let us see various criterion to decide A grade for ESC101 ☺

Values generated by Logical Ops

a = 1 if good marks, else 0, b = 1 if good attendance, else 0

Let us see various criterion to decide A grade for ESC101 ☺

Operation	C Code	a	b
		0	0
		0	1
		1	0
		1	1

Values generated by Logical Ops

a = 1 if good marks, else 0, b = 1 if good attendance, else 0

Let us see various criterion to decide A grade for ESC101 ☺

Operation	C Code	a	b
		0	0
		0	1
		1	0
		1	1

Values generated by Logical Ops

a = 1 if good marks, else 0, b = 1 if good attendance, else 0

Let us see various criterion to decide A grade for ESC101 ☺

Operation	C Code	a	b
		0	0
		0	1
		1	0
		1	1

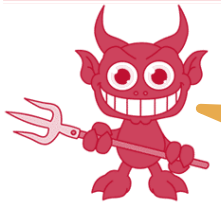
A if good attendance and good marks

Values generated by Logical Ops

a = 1 if good marks, else 0, b = 1 if good attendance, else 0

Let us see various criterion to decide A grade for ESC101 ☺

Operation	C Code	a	b
AND	c = a && b	0	0
		0	1
		1	0
		1	1



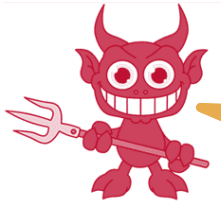
A if good attendance and good marks

Values generated by Logical Ops

a = 1 if good marks, else 0, b = 1 if good attendance, else 0

Let us see various criterion to decide A grade for ESC101 ☺

Operation	C Code	a	b	c
AND	**c = a && b**	**0**	**0**	**0**
		0	**1**	**0**
		1	**0**	**0**
		1	**1**	**1**



A if good attendance and good marks

Values generated by Logical Ops

a = 1 if good marks, else 0, b = 1 if good attendance, else 0

Let us see various criterion to decide A grade for ESC101 ☺

Operation	C Code	a	b	c
AND	c = a && b	0	0	0
		0	1	0
		1	0	0
		1	1	1

Values generated by Logical Ops

a = 1 if good marks, else 0, b = 1 if good attendance, else 0

Let us see various criterion to decide A grade for ESC101 ☺

Operation	C Code	a	b	c
AND	c = a && b	0	0	0
		0	1	0
		1	0	0
		1	1	1

Values generated by Logical Ops

a = 1 if good marks, else 0, b = 1 if good attendance, else 0

Let us see various criterion to decide A grade for ESC101 ☺

Operation	C Code	a	b	c
AND	c = a && b	0	0	0
		0	1	0
		1	0	0
		1	1	1

A if good attendance or good marks or both

Values generated by Logical Ops

a = 1 if good marks, else 0, b = 1 if good attendance, else 0

Let us see various criterion to decide A grade for ESC101 ☺

Operation	C Code	a	b	c
AND	c = a && b	0	0	0
OR	d = a || b	0	1	0
		1	0	0
		1	1	1

A if good attendance or good marks or both

Values generated by Logical Ops

a = 1 if good marks, else 0, b = 1 if good attendance, else 0

Let us see various criterion to decide A grade for ESC101 ☺

Operation	C Code	a	b	c	d
AND	c = a && b	0	0	0	0
OR	d = a || b	0	1	0	1
		1	0	0	1
		1	1	1	1

A if good attendance or good marks or both

Values generated by Logical Ops

a = 1 if good marks, else 0, b = 1 if good attendance, else 0

Let us see various criterion to decide A grade for ESC101 ☺

Operation	C Code	a	b	c	d
AND	c = a && b	0	0	0	0
OR	d = a || b	0	1	0	1
		1	0	0	1
		1	1	1	1

Values generated by Logical Ops

a = 1 if good marks, else 0, b = 1 if good attendance, else 0

Let us see various criterion to decide A grade for ESC101 ☺

Operation	C Code	a	b	c	d
AND	c = a && b	0	0	0	0
OR	d = a || b	0	1	0	1
		1	0	0	1
		1	1	1	1

Values generated by Logical Ops

a = 1 if good marks, else 0, b = 1 if good attendance, else 0

Let us see various criterion to decide A grade for ESC101 ☺

Operation	C Code	a	b	c	d
AND	c = a && b	0	0	0	0
OR	d = a || b	0	1	0	1
		1	0	0	1
		1	1	1	1

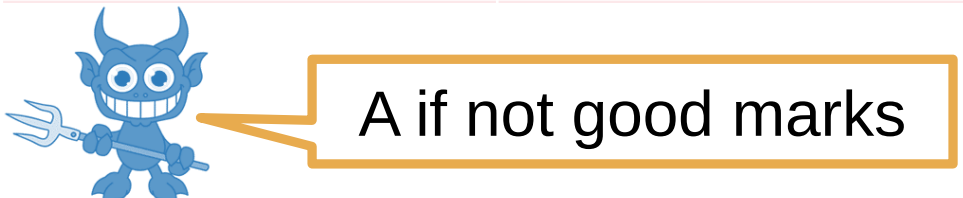
A if not good marks

Values generated by Logical Ops

a = 1 if good marks, else 0, b = 1 if good attendance, else 0

Let us see various criterion to decide A grade for ESC101 ☺

Operation	C Code	a	b	c	d
AND	c = a && b	0	0	0	0
OR	d = a || b	0	1	0	1
NOT	e = !a	1	0	0	1
		1	1	1	1



Values generated by Logical Ops

a = 1 if good marks, else 0, b = 1 if good attendance, else 0

Let us see various criterion to decide A grade for ESC101 ☺

Operation	C Code	a	b	c	d	e
AND	c = a && b	0	0	0	0	1
OR	d = a || b	0	1	0	1	1
NOT	e = !a	1	0	0	1	0
		1	1	1	1	0

A if not good marks

Values generated by Logical Ops

a = 1 if good marks, else 0, b = 1 if good attendance, else 0

Let us see various criterion to decide A grade for ESC101 ☺

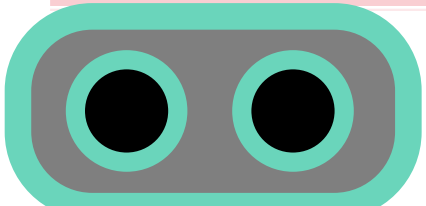
Operation	C Code	a	b	c	d	e
AND	c = a && b	0	0	0	0	1
OR	d = a || b	0	1	0	1	1
NOT	e = !a	1	0	0	1	0
		1	1	1	1	0

Values generated by Logical Ops

a = 1 if good marks, else 0, b = 1 if good attendance, else 0

Let us see various criterion to decide A grade for ESC101 ☺

Operation	C Code	a	b	c	d	e
AND	c = a && b	0	0	0	0	1
OR	d = a || b	0	1	0	1	1
NOT	e = !a	1	0	0	1	0
		1	1	1	1	0

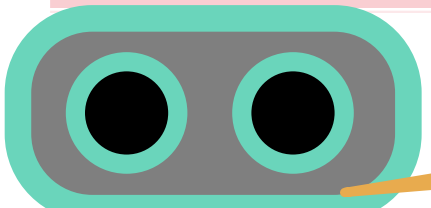


Values generated by Logical Ops

a = 1 if good marks, else 0, b = 1 if good attendance, else 0

Let us see various criterion to decide A grade for ESC101 ☺

Operation	C Code	a	b	c	d	e
AND	c = a && b	0	0	0	0	1
OR	d = a || b	0	1	0	1	1
NOT	e = !a	1	0	0	1	0
					1	0



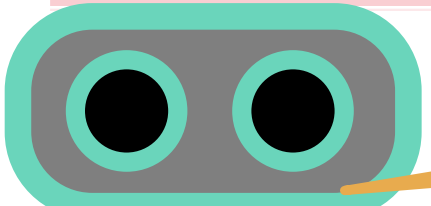
Challenge: write a relational expression which evaluates to 1 only if exactly one of a and b is 1

Values generated by Logical Ops

a = 1 if good marks, else 0, b = 1 if good attendance, else 0

Let us see various criterion to decide A grade for ESC101 ☺

Operation	C Code	a	b	c	d	e
AND	c = a && b	0	0	0	0	1
OR	d = a || b	0	1	0	1	1
NOT	e = !a	1	0	0	1	0
					1	0

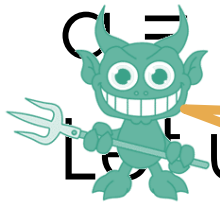


Challenge: write a relational expression which evaluates to 1 only if exactly one of a and b is 1

Values generated by Logical Ops

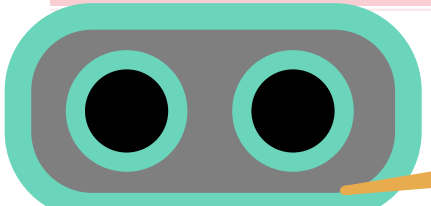
c = 1 if ... 0, b = 1 if good attendance, else 0

Let us ... on to decide A grade for ESC101 ☺



A if good attendance or good marks but not both

Operation	C Code	a	b	c	d	e
AND	c = a && b	0	0	0	0	1
OR	d = a || b	0	1	0	1	1
NOT	e = !a	1	0	0	1	0
					1	0

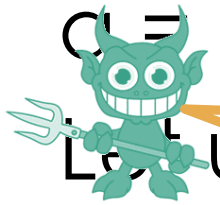


Challenge: write a relational expression which evaluates to 1 only if exactly one of a and b is 1

Values generated by Logical Ops

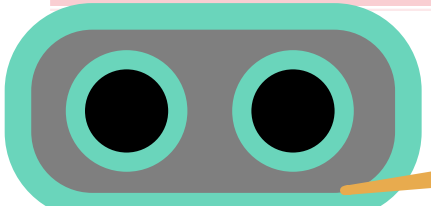
a = 1 if [illegible] 0, b = 1 if good attendance, else 0

Let us [illegible]on to decide A grade for ESC101 ☺

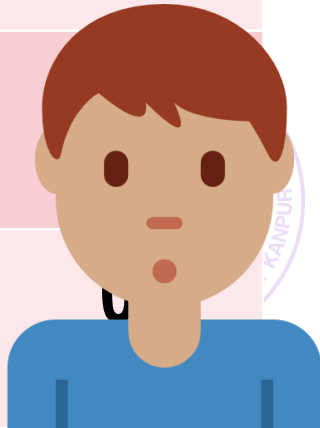


A if good attendance or good marks but not both

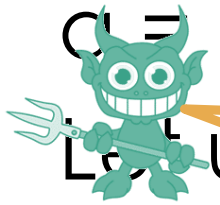
Operation	C Code	a	b	c	d	e
AND	c = a && b	0	0	0	0	1
OR	d = a || b	0	1	0	1	1
NOT	e = !a	1	0	0	1	[illegible]
					1	[illegible]



Challenge: write a relational expression which evaluates to 1 only if exactly one of a and b is 1



Values generated by Logical Ops



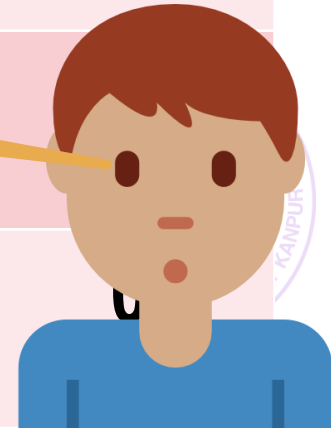
A if good attendance or good marks but not both

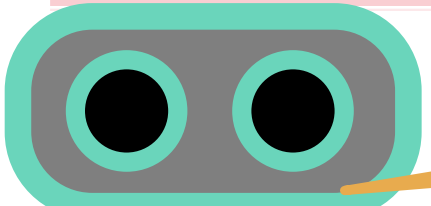
a = 1 if [illegible] 0, b = 1 if good attendance, else 0

Let us [illegible]on to decide A grade for ESC101 ☺

Operation	C Code	a	b	c	d	e
AND	c = a && b	0	0	0	0	1
OR	d = a || b	0	1	0	1	1
NOT	e = !a	1	[illegible]	[illegible]	[illegible]	[illegible]
					1	[illegible]

So if a = 1 and b = 0 then the answer should be 1



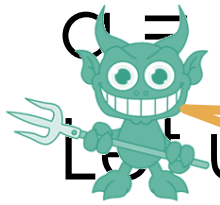


Challenge: write a relational expression which evaluates to 1 only if exactly one of a and b is 1

Values generated by Logical Ops

c = 1 if [illegible] 0, b = 1 if good attendance, else 0

Let us [illegible]on to decide A grade for ESC101 ☺

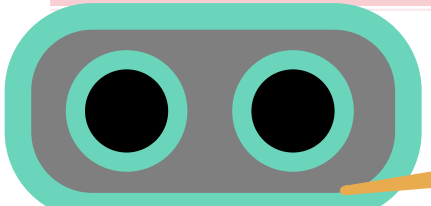


A if good attendance or good marks but not both

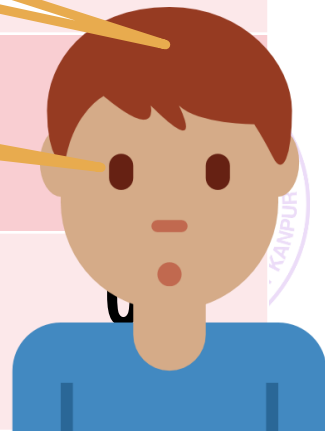
Operation	C Code	a	b	c	d	e
AND	c = a && b	0	0	0	0	1
OR	d = a || b	0				1
NOT	e = !a	1				
					1	

But if a = 1, b = 1, then answer should be 0

So if a = 1 and b = 0 then the answer should be 1



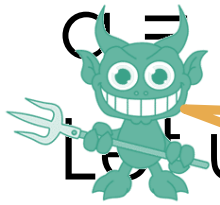
Challenge: write a relational expression which evaluates to 1 only if exactly one of a and b is 1



Values generated by Logical Ops

a = 1 if [illegible] 0, b = 1 if good attendance, else 0

Let us [illegible]on to decide A grade for ESC101 ☺



A if good attendance or good marks but not both

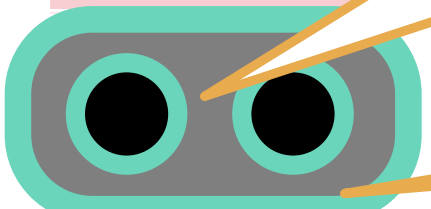
Operation	C Code	a	b	c	d	e
AND	c = a && b	0	0	0	0	1
OR	d = a || b	0				1
NO[illegible]		[illegible]1				
					1	

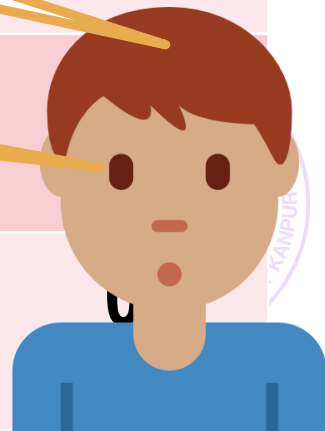
But if a = 1, b = 1, then answer should be 0

So if a = 1 and b = 0 then the answer should be 1

Correct! This is called the XOR (exclusive or) operation

Challenge: write a relational expression which evaluates to 1 only if exactly one of a and b is 1

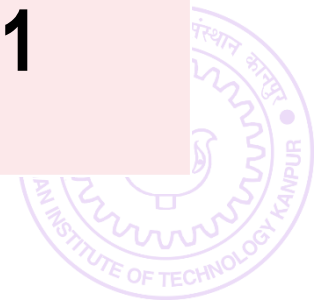




Some Examples

Some Examples

C Code	a	b	c	d	e
c = (a && b) || a	0	0	0	1	0
d = !(a && b)	0	1	0	1	0
e = (a || b) && !d	1	0	1	1	0
	1	1	1	0	1



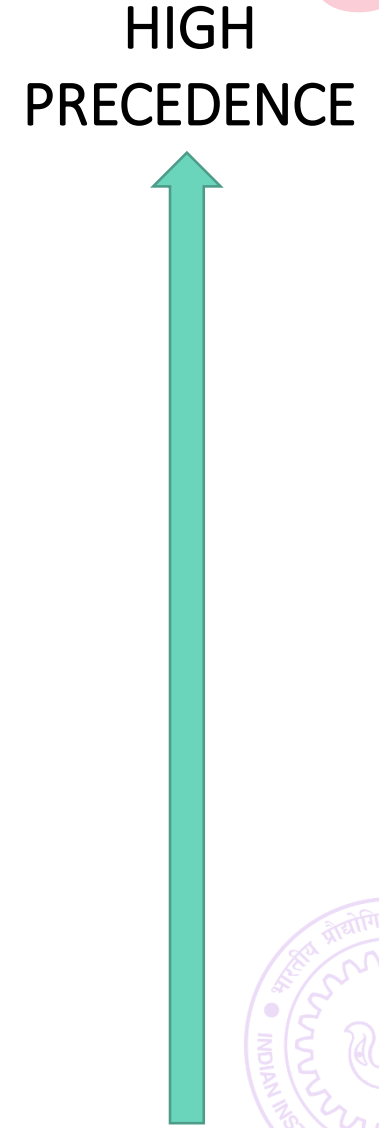
BODMAS table has more members

BODMAS table has more members

Operator Name	Symbol/Sign	Associativity
Bracket, Post increment/decrement	(), ++, --	Left
Unary negation, Pre increment/decrement, **NOT**	-, ++, --, **!**	Right
Multiplication/division/ remainder	*, /, %	Left
Addition/subtraction	+, -	Left
Relational	**<, <=, >, >=**	Left
Relational	**==, !=**	Left
AND	**&&**	Left
OR	**||**	Left
Assignment, Compound assignment	=, +=, -=, *=, /=, %=	Right

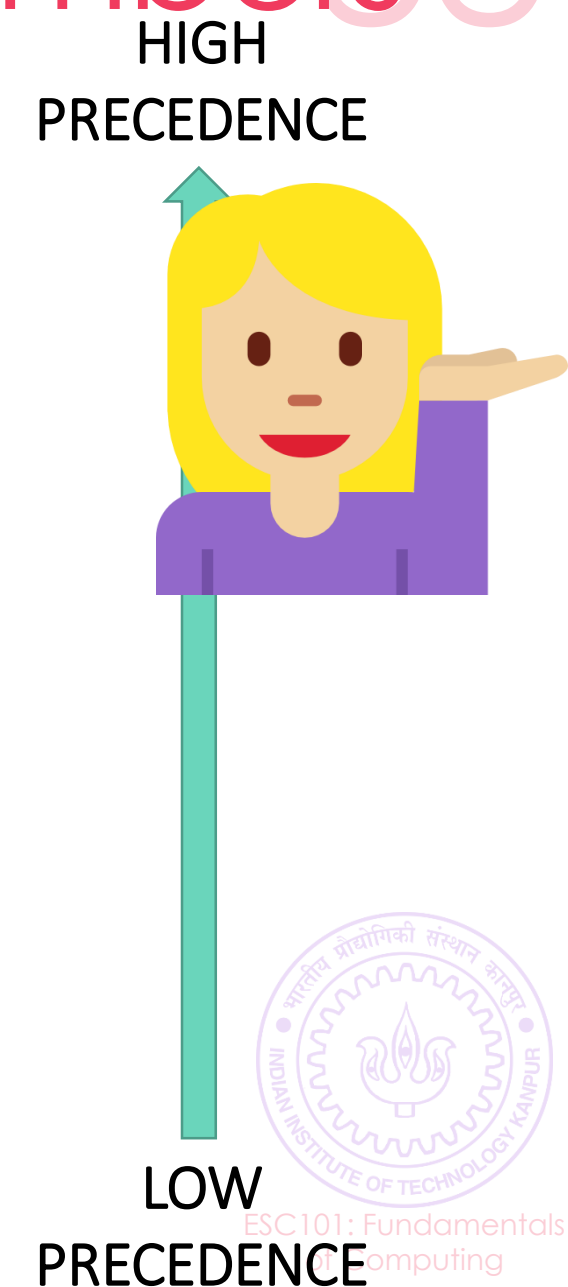
BODMAS table has more members

Operator Name	Symbol/Sign	Associativity
Bracket, Post increment/decrement	(), ++, --	Left
Unary negation, Pre increment/decrement, **NOT**	-, ++, --, **!**	Right
Multiplication/division/ remainder	*, /, %	Left
Addition/subtraction	+, -	Left
Relational	**<, <=, >, >=**	Left
Relational	**==, !=**	Left
AND	**&&**	Left
OR	**||**	Left
Assignment, Compound assignment	=, +=, -=, *=, /=, %=	Right



BODMAS table has more members

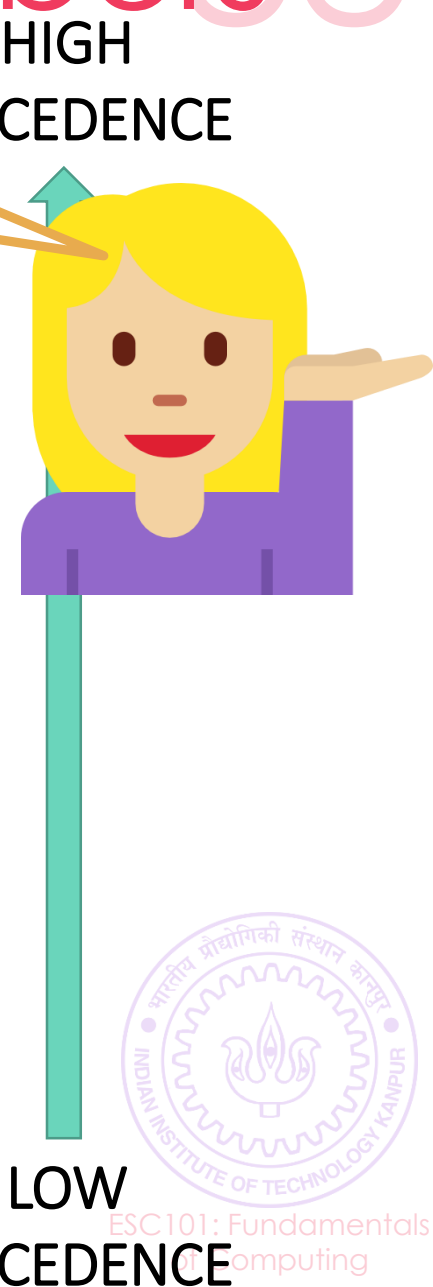
Operator Name	Symbol/Sign	Associativity
Bracket, Post increment/decrement	(), ++, --	Left
Unary negation, Pre increment/decrement, **NOT**	-, ++, --, **!**	Right
Multiplication/division/ remainder	*, /, %	Left
Addition/subtraction	+, -	Left
Relational	**<, <=, >, >=**	Left
Relational	**==, !=**	Left
AND	**&&**	Left
OR	**||**	Left
Assignment, Compound assignment	=, +=, -=, *=, /=, %=	Right



BODMAS table has more members

Operator Name		
Bracket, Post increment/decrement		
Unary negation, Pre increment/decrement, **NOT**	-, ++, --, **!**	Right
Multiplication/division/ remainder	*, /, %	Left
Addition/subtraction	+, -	Left
Relational	**<, <=, >, >=**	Left
Relational	**==, !=**	Left
AND	**&&**	Left
OR	**||**	Left
Assignment, Compound assignment	=, +=, -=, *=, /=, %=	Right

Write this table down in your notebook. Allowed in labs, quizzes, exams. No need to memorize.



Some Fun with Operators

Some Fun with Operators

Given three integers a, b, c, find how many are even

Some Fun with Operators

Given three integers a, b, c, find how many are even

```
(a % 2 == 0) + (b % 2 == 0) + (c % 2 == 0)
```

Some Fun with Operators

Given three integers a, b, c, find how many are even

```
(a % 2 == 0) + (b % 2 == 0) + (c % 2 == 0)
```

Some Fun with Operators

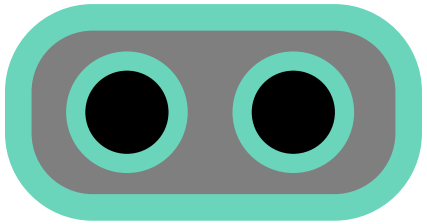
Given three integers a, b, c, find how many are even

$$(a \% 2 == 0) + (b \% 2 == 0) + (c \% 2 == 0)$$

Some Fun with Operators

Given three integers a, b, c, find how many are even

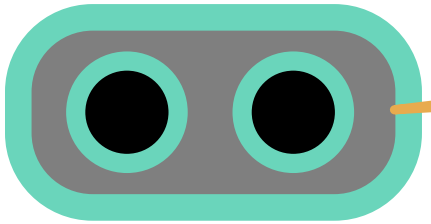
$$(a \% 2 == 0) + (b \% 2 == 0) + (c \% 2 == 0)$$



Some Fun with Operators

Given three integers a, b, c, find how many are even

```
(a % 2 == 0) + (b % 2 == 0) + (c % 2 == 0)
```



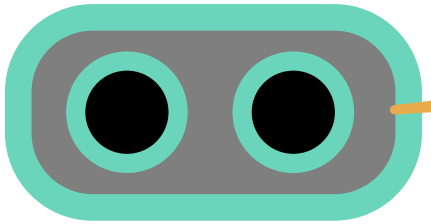
Works because relational expressions generate values which are 0 or 1

Some Fun with Operators

Given three integers a, b, c, find how many are even

```
(a % 2 == 0) + (b % 2 == 0) + (c % 2 == 0)
```

Common mistakes



Works because relational expressions generate values which are 0 or 1

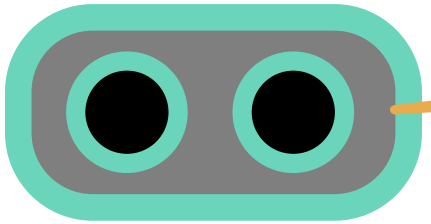
Some Fun with Operators

Given three integers a, b, c, find how many are even

```
(a % 2 == 0) + (b % 2 == 0) + (c % 2 == 0)
```

Common mistakes

```
if(b = 3){ printf("Hello"); }
```



Works because relational expressions generate values which are 0 or 1

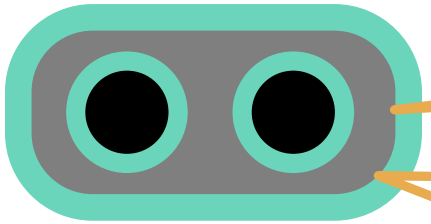
Some Fun with Operators

Given three integers a, b, c, find how many are even

```
(a % 2 == 0) + (b % 2 == 0) + (c % 2 == 0)
```

Common mistakes

```
if(b = 3){ printf("Hello"); }
```



Works because relational expressions generate values which are 0 or 1

I will always print Hello no matter what the value of b

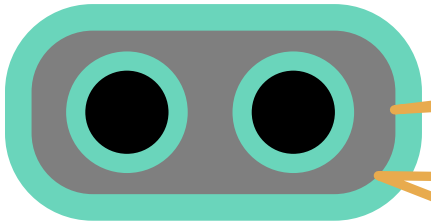
Some Fun with Operators

Given three integers a, b, c, find how many are even

```
(a % 2 == 0) + (b % 2 == 0) + (c % 2 == 0)
```

Common mistakes

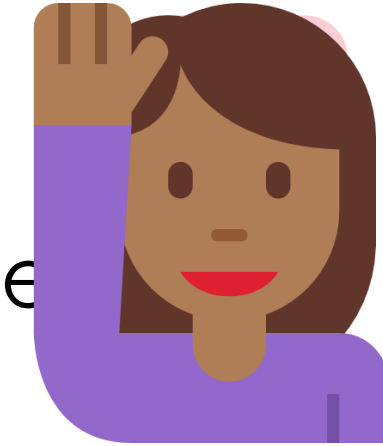
```
if(b = 3){ printf("Hello"); }
```



Works because relational expressions generate values which are 0 or 1

I will always print Hello no matter what the value of b

Some Fun with Operators

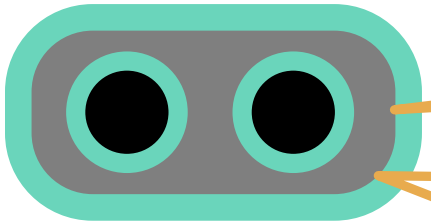


Given three integers a, b, c, find how many are eve

```
(a % 2 == 0) + (b % 2 == 0) + (c % 2 == 0)
```

Common mistakes

```
if(b = 3){ printf("Hello"); }
```



Works because relational expressions generate values which are 0 or 1

I will always print Hello no matter what the value of b

Some Fun with Operators

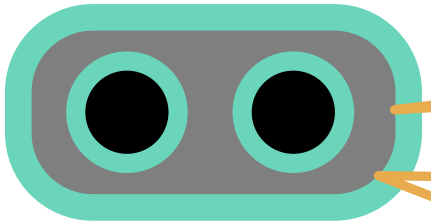
Given three integers a, b

(a % 2 == 0) + (b %

Common mistakes

if(b = 3){ printf("Hello"); }

Mr C considers 0 to be FALSE and 1 (or anything non-zero) to be TRUE



Works because relational expressions generate values which are 0 or 1

I will always print Hello no matter what the value of b

Wow!

Why?!

Some Fun with Operators

Given three integers a, b, c, find if

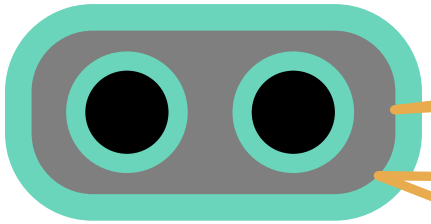
(a % 2 == 0) + (b %

Common mistakes

if(b = 3){ printf("Hello"); }

Mr C considers 0 to be FALSE and 1 (or anything non-zero) to be TRUE

b = 3 generates the value 3 which is non-zero. So Mr. C will consider it to be true



Works because relational expressions generate values which are 0 or 1

I will always print Hello no matter what the value of b

Wow!

Why?!

Some Fun with Operators

Given three integers a, b, c, find if

(a % 2 == 0) + (b %

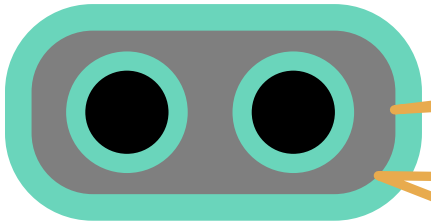
Common mistakes

if(b = 3){ printf("Hello"); }

Common tricks used by more flamboyant coders

Mr C considers 0 to be FALSE and 1 (or anything non-zero) to be TRUE

b = 3 generates the value 3 which is non-zero. So Mr. C will consider it to be true



Works because relational expressions generate values which are 0 or 1

I will always print Hello no matter what the value of b

Wow!

Why?!

Some Fun with Operators

Given three integers a, b, c, find if

(a % 2 == 0) + (b %

Common mistakes

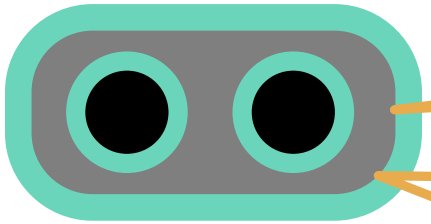
if(b = 3){ printf("Hello"); }

Common tricks used by more flamboyant coders

if(1){ printf("Bye"); }

Mr C considers 0 to be FALSE and 1 (or anything non-zero) to be TRUE

b = 3 generates the value 3 which is non-zero. So Mr. C will consider it to be true



Works because relational expressions generate values which are 0 or 1

I will always print Hello no matter what the value of b

Wow!

Why?!

Some Fun with Operators

Given three integers a, b

(a % 2 == 0) + (b %

Common mistakes

if(b = 3){ printf("Hello"); }

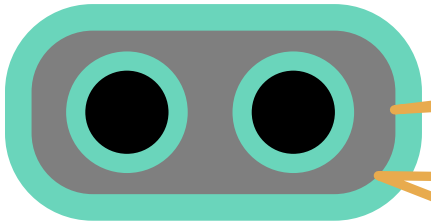
Common tricks used by more flamboyant coders

if(1){ printf("Bye"); }

Mr C considers 0 to be FALSE and 1 (or anything non-zero) to be TRUE

b = 3 generates the value 3 which is non-zero. So Mr. C will consider it to be true

Will always print Bye



Works because relational expressions generate values which are 0 or 1

I will always print Hello no matter what the value of b

Wow!

Why?!

Some Fun with Operators

Given three integers a, b

(a % 2 == 0) + (b %

Common mistakes

if(b = 3){ printf("Hello"); }

Common tricks used by more flamboyant coders

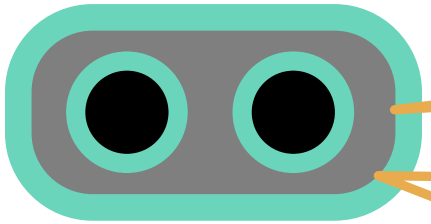
if(1){ printf("Bye"); }

Mr C considers 0 to be FALSE and 1 (or anything non-zero) to be TRUE

b = 3 generates the value 3 which is non-zero. So Mr. C will consider it to be true

Will always print Bye

if(10) and if(-25.6) also do the same



Works because relational expressions generate values which are 0 or 1

I will always print Hello no matter what the value of b

Wow!

Why?!

Some Fun with Operators

Given three integers a, b

```
(a % 2 == 0) + (b %
```

Common mistakes

```
if(b = 3){ printf("Hello"); }
```

Common tricks used by more flamboyant coders

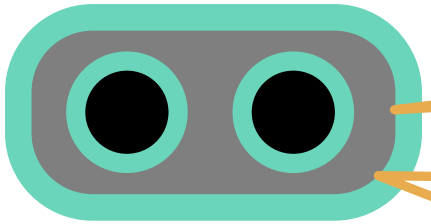
```
if(1){ printf("Bye"); }
if(0){ printf("Hi"); }
```

Mr C considers 0 to be FALSE and 1 (or anything non-zero) to be TRUE

b = 3 generates the value 3 which is non-zero. So Mr. C will consider it to be true

Will always print Bye

if(10) and if(-25.6) also do the same



Works because relational expressions generate values which are 0 or 1

I will always print Hello no matter what the value of b

Wow!

Why?!

Some Fun with Operators

Given three integers a, b

```
(a % 2 == 0) + (b %
```

Common mistakes

```
if(b = 3){ printf("Hello"); }
```

Common tricks used by more flamboyant coders

```
if(1){ printf("Bye"); }
```

```
if(0){ printf("Hi"); }
```

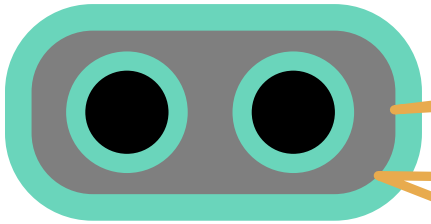
Mr C considers 0 to be FALSE and 1 (or anything non-zero) to be TRUE

b = 3 generates the value 3 which is non-zero. So Mr. C will consider it to be true

Will always print Bye

if(10) and if(-25.6) also do the same

Will never print Hi



Works because relational expressions generate values which are 0 or 1

I will always print Hello no matter what the value of b

Wow!

Why?!

Practice Problems

Practice Problems

Will the following statements evaluate to true or false?

Practice Problems

Will the following statements evaluate to true or false?

Remember 0 is false and non-zero (e.g. 1, 10.0) is true

Practice Problems

Will the following statements evaluate to true or false?

Remember 0 is false and non-zero (e.g. 1, 10.0) is true

```
int a = 5, b = 6, c = 7;
```

Practice Problems

Will the following statements evaluate to true or false?

Remember 0 is false and non-zero (e.g. 1, 10.0) is true

```
int a = 5, b = 6, c = 7;
```

```
a > b == c
```

Practice Problems

Will the following statements evaluate to true or false?

Remember 0 is false and non-zero (e.g. 1, 10.0) is true

```
int a = 5, b = 6, c = 7;
```

```
a > b == c
```

```
a = a – 5 > 4
```

Practice Problems

Will the following statements evaluate to true or false?

Remember 0 is false and non-zero (e.g. 1, 10.0) is true

```
int a = 5, b = 6, c = 7;
```

```
a > b == c
```

```
a = a – 5 > 4
```

```
c -- == 0
```

Practice Problems

Will the following statements evaluate to true or false?

Remember 0 is false and non-zero (e.g. 1, 10.0) is true

```
int a = 5, b = 6, c = 7;
```

```
a > b == c
```

```
a = a – 5 > 4
```

```
c -- == 0
```

Practice Problems

Will the following statements evaluate to true or false?

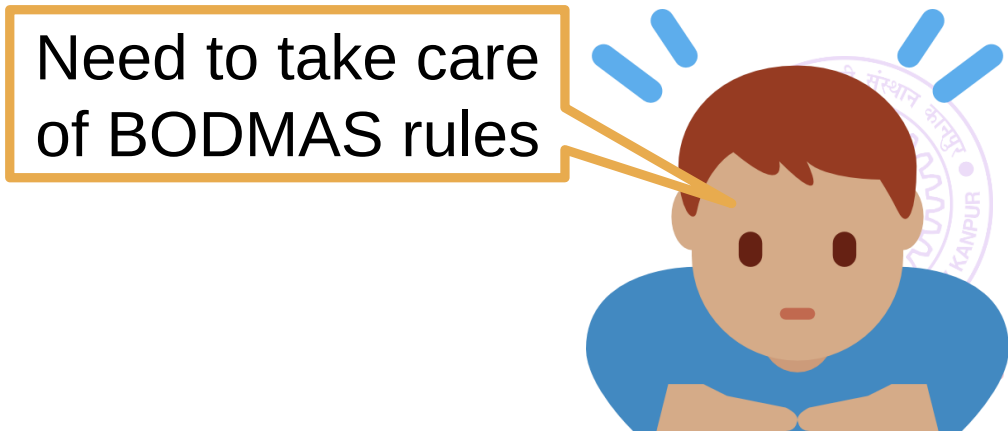
Remember 0 is false and non-zero (e.g. 1, 10.0) is true

```
int a = 5, b = 6, c = 7;
```

```
a > b == c
```

```
a = a – 5 > 4
```

```
c -- == 0
```



Practice Problems

Will the following statements evaluate to true or false?

Remember 0 is false and non-zero (e.g. 1, 10.0) is true

```
int a = 5, b = 6, c = 7;
```

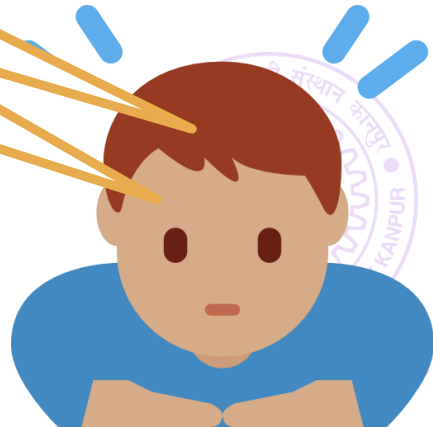
```
a > b == c
```

```
a = a – 5 > 4
```

```
c -- == 0
```

Use these expressions in a printf statement to check your answers

Need to take care of BODMAS rules



A few handy shortcuts

A few handy shortcuts

If we want Mr C to do only one thing (i.e. single statement) if something happens, then brackets not necessary

A few handy shortcuts

If we want Mr C to do only one thing (i.e. single statement) if something happens, then brackets not necessary

A few handy shortcuts

If we want Mr C to do only one thing (i.e. single statement) if something happens, then brackets not necessary

```
if(sum < 10){
```

A few handy shortcuts

If we want Mr C to do only one thing (i.e. single statement) if something happens, then brackets not necessary

```
if(sum < 10){
    printf("Small");
```

A few handy shortcuts

If we want Mr C to do only one thing (i.e. single statement) if something happens, then brackets not necessary

```
if(sum < 10){
    printf("Small");
}else{
```

A few handy shortcuts

If we want Mr C to do only one thing (i.e. single statement) if something happens, then brackets not necessary

```
if(sum < 10){
    printf("Small");
}else{
    printf("Big");
```

A few handy shortcuts

If we want Mr C to do only one thing (i.e. single statement) if something happens, then brackets not necessary

```
if(sum < 10){
    printf("Small");
}else{
    printf("Big");
}
```

A few handy shortcuts

If we want Mr C to do only one thing (i.e. single statement) if something happens, then brackets not necessary

```
if(sum < 10){
    printf("Small");
}else{
    printf("Big");
}
printf("Goodbye");
```

A few handy shortcuts

If we want Mr C to do only one thing (i.e. single statement) if something happens, then brackets not necessary

```
if(sum < 10)
    printf("Small");
else
    printf("Big");
printf("Goodbye");
```

A few handy shortcuts

If we want Mr C to do only one thing (i.e. single statement) if something happens, then brackets not necessary

Some programmers write that statement on same line

```
if(sum < 10)
    printf("Small");
else
    printf("Big");
printf("Goodbye");
```

A few handy shortcuts

If we want Mr C to do only one thing (i.e. single statement) if something happens, then brackets not necessary

Some programmers write that statement on same line

```
if(sum < 10) printf("Small");
else printf("Big");
printf("Goodbye");
```

A few handy shortcuts

If we want Mr C to do only one thing (i.e. single statement) if something happens, then brackets not necessary

Some programmers write that statement on same line

Still a good idea to indent

```
if(sum < 10) printf("Small");
else printf("Big");
printf("Goodbye");
```

A few handy shortcuts

If we want Mr C to do only one thing (i.e. single statement) if something happens, then brackets not necessary

Some programmers write that statement on same line

Still a good idea to indent

```
if(sum < 10) printf("Small");
else printf("Big");
printf("Goodbye");
```

```
if(sum < 10)
    printf("Small");
else
    printf("Big");
printf("Goodbye");
```

Risky business with brackets

Sometimes, to make code look pretty, professional programmers omit brackets when not needed

Risky business with brackets

Sometimes, to make code look pretty, professional programmers omit brackets when not needed

Risky business with brackets

Sometimes, to make code look pretty, professional programmers omit brackets when not needed

```
if(a == 1) printf(“One”);
```

Risky business with brackets

Sometimes, to make code look pretty, professional programmers omit brackets when not needed

```
if(a == 1) printf("One");
else if(a == 2) printf("Two");
```

Risky business with brackets

Sometimes, to make code look pretty, professional programmers omit brackets when not needed

```
if(a == 1) printf("One");
else if(a == 2) printf("Two");
else if(a == 3) printf("Three");
```

Risky business with brackets

Sometimes, to make code look pretty, professional programmers omit brackets when not needed

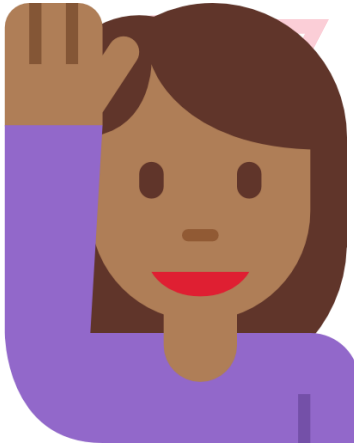
```
if(a == 1) printf("One");
else if(a == 2) printf("Two");
else if(a == 3) printf("Three");
else if(a == 4) printf("Four");
```

Risky business with brackets

Sometimes, to make code look pretty, professional programmers omit brackets when not needed

```
if(a == 1)
    printf(“One”);
else if(a == 2)
    printf(“Two”);
else if(a == 3)
    printf(“Three”);
else if(a == 4)
    printf(“Four”);
```

Risky business with brackets

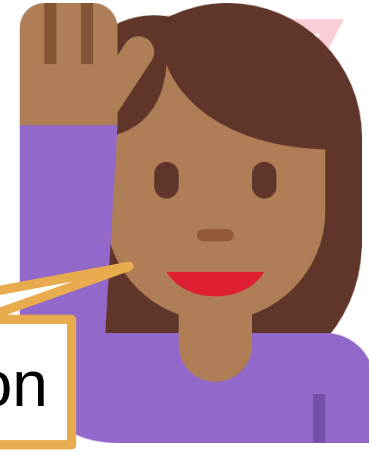


Sometimes, to make code look pretty, professional programmers omit brackets when not needed

```
if(a == 1)
    printf(“One”);
else if(a == 2)
    printf(“Two”);
else if(a == 3)
    printf(“Three”);
else if(a == 4)
    printf(“Four”);
```

Risky business with brackets

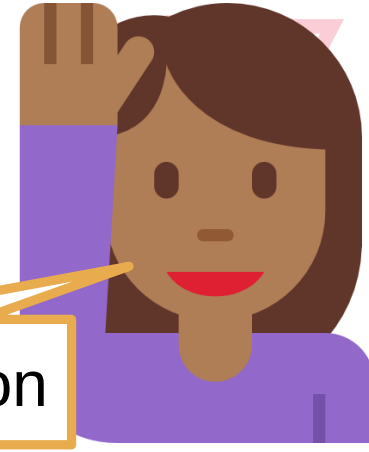
Sometimes, to make code look pretty, professional programmers omit brackets when not r




```
if(a == 1)
    printf("One");
else if(a == 2)
    printf("Two");
else if(a == 3)
    printf("Three");
else if(a == 4)
    printf("Four");
```

Risky business with brackets

Sometimes, to make code look pretty, professional programmers omit brackets when not r

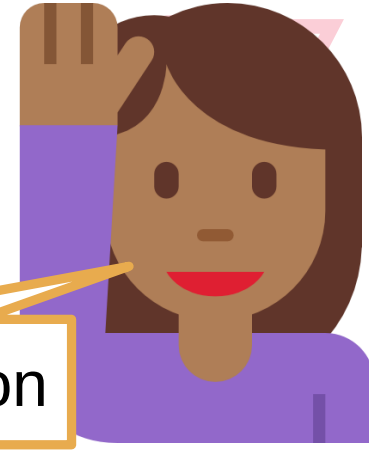



```
if(a == 1)
    printf("One");
else if(a == 2)
    printf("Two");
else if(a == 3)
    printf("Three");
else if(a == 4)
    printf("Four");
```

SAME OUTPUT

Risky business with brackets

Sometimes, to make code look pretty, professional programmers omit brackets when not r

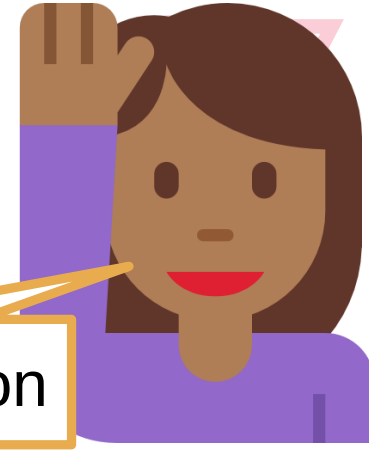



```
if(a == 1)
    printf("One");
else if(a == 2)
    printf("Two");
else if(a == 3)
    printf("Three");
else if(a == 4)
    printf("Four");
```

SAME OUTPUT

Risky business with brackets

Sometimes, to make code look pretty, professional programmers omit brackets when not r



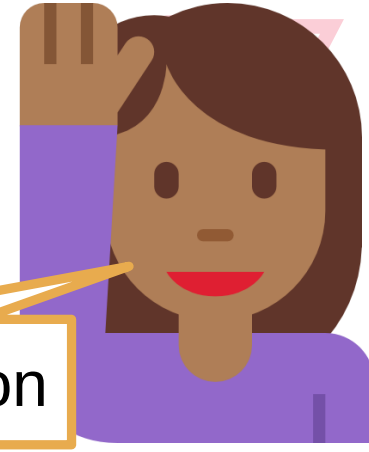

```
if(a == 1)
    printf("One");
else if(a == 2)
    printf("Two");
else if(a == 3)
    printf("Three");
else if(a == 4)
    printf("Four");
```

SAME OUTPUT

```
if(a == 1){


}else{









}
```

Risky business with brackets

Sometimes, to make code look pretty, professional programmers omit brackets when not r



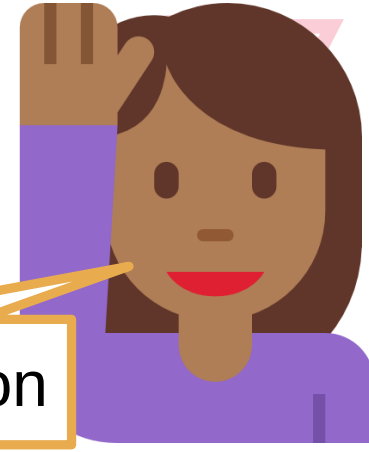

```
if(a == 1)
    printf("One");
else if(a == 2)
    printf("Two");
else if(a == 3)
    printf("Three");
else if(a == 4)
    printf("Four");
```

SAME OUTPUT

```
if(a == 1){
    printf("One");
}else{
                                                    }
```

Risky business with brackets

Sometimes, to make code look pretty, professional programmers omit brackets when not r



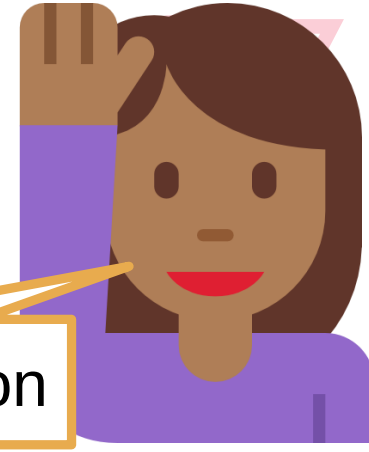

```
if(a == 1)
    printf("One");
else if(a == 2)
    printf("Two");
else if(a == 3)
    printf("Three");
else if(a == 4)
    printf("Four");
```

SAME OUTPUT

```
if(a == 1){
    printf("One");
}else{ if(a == 2){

        }else{

                                    }}
```

Risky business with brackets

Sometimes, to make code look pretty, professional programmers omit brackets when not r



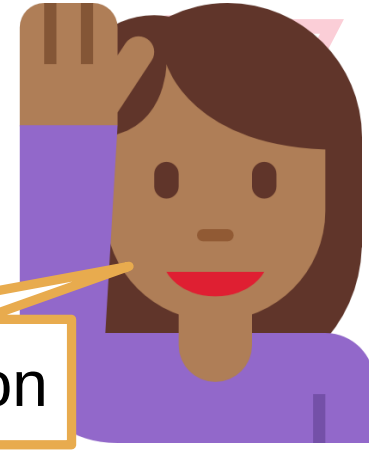

```
if(a == 1)
    printf("One");
else if(a == 2)
    printf("Two");
else if(a == 3)
    printf("Three");
else if(a == 4)
    printf("Four");
```

SAME OUTPUT

```
if(a == 1){
    printf("One");
}else{ if(a == 2){
        printf("Two");
    }else{

                                    }}
```

Risky business with brackets

Sometimes, to make code look pretty, professional programmers omit brackets when not r



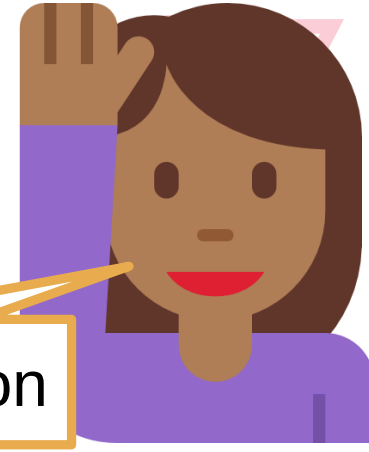

```
if(a == 1)
    printf("One");
else if(a == 2)
    printf("Two");
else if(a == 3)
    printf("Three");
else if(a == 4)
    printf("Four");
```

SAME OUTPUT

```
if(a == 1){
    printf("One");
}else{ if(a == 2){
        printf("Two");
    }else{ if(a == 3){

            }else{

                                }}}
```

Risky business with brackets

Sometimes, to make code look pretty, professional programmers omit brackets when not r



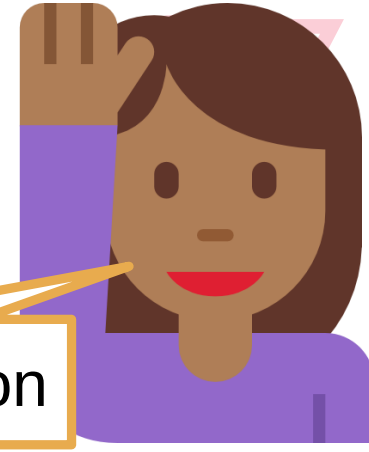

```
if(a == 1)
    printf("One");
else if(a == 2)
    printf("Two");
else if(a == 3)
    printf("Three");
else if(a == 4)
    printf("Four");
```

SAME OUTPUT

```
if(a == 1){
    printf("One");
}else{ if(a == 2){
        printf("Two");
    }else{ if(a == 3){
            printf("Three");
        }else{
                                }}}
```

Risky business with brackets

Sometimes, to make code look pretty, professional programmers omit brackets when not r



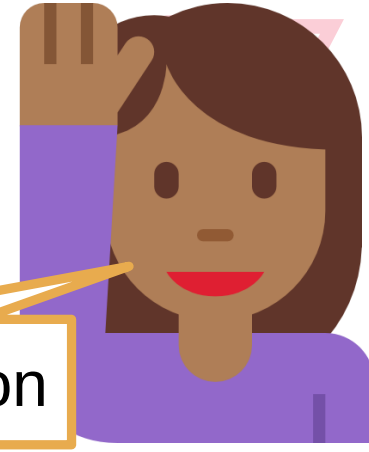

```
if(a == 1)
    printf("One");
else if(a == 2)
    printf("Two");
else if(a == 3)
    printf("Three");
else if(a == 4)
    printf("Four");
```

SAME OUTPUT

```
if(a == 1){
    printf("One");
}else{ if(a == 2){
        printf("Two");
    }else{ if(a == 3){
            printf("Three");
        }else{ if(a == 4){
                }}}}
```

Risky business with brackets

Sometimes, to make code look pretty, professional programmers omit brackets when not r

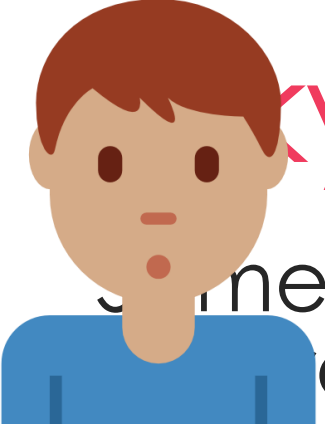



```
if(a == 1)
    printf("One");
else if(a == 2)
    printf("Two");
else if(a == 3)
    printf("Three");
else if(a == 4)
    printf("Four");
```

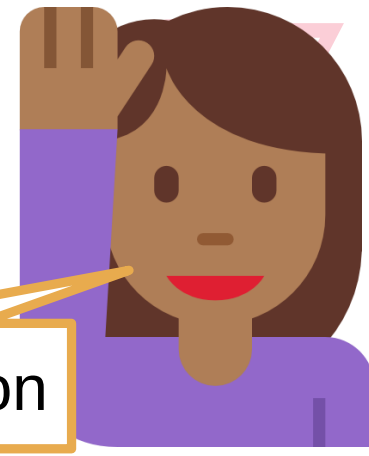
SAME OUTPUT

```
if(a == 1){
    printf("One");
}else{ if(a == 2){
        printf("Two");
    }else{ if(a == 3){
            printf("Three");
        }else{ if(a == 4){
                printf("Four");}}}}
```

ky business with brackets



netimes, to make code look pretty, professional
ammers omit brackets when not r

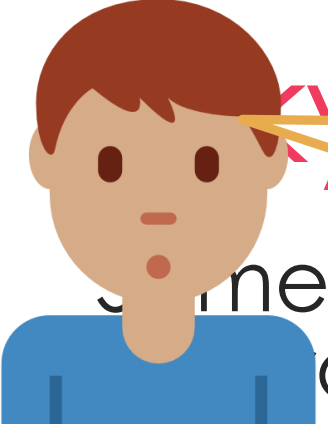



```
if(a == 1)
    printf(“One”);
else if(a == 2)
    printf(“Two”);
else if(a == 3)
    printf(“Three”);
else if(a == 4)
    printf(“Four”);
```

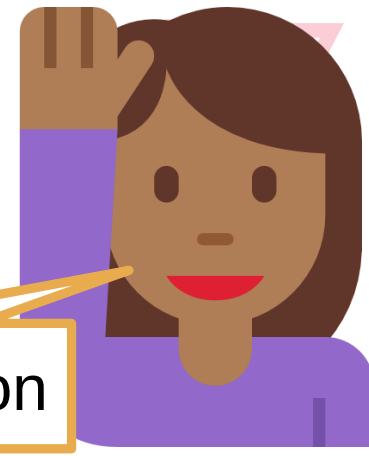
SAME OUTPUT

```
if(a == 1){
    printf(“One”);
}else{ if(a == 2){
        printf(“Two”);
    }else{ if(a == 3){
            printf(“Three”);
        }else{ if(a == 4){
                printf(“Four”);}}}}
```

...rackets



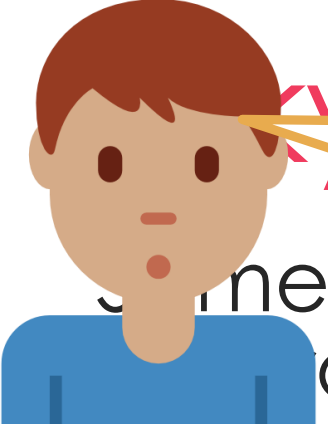
...metimes, to make code look pretty, professional ...ammers omit brackets when not r...




```
if(a == 1)
    printf("One");
else if(a == 2)
    printf("Two");
else if(a == 3)
    printf("Three");
else if(a == 4)
    printf("Four");
```

SAME OUTPUT

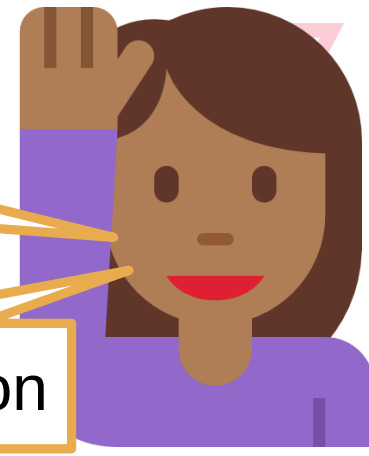
```
if(a == 1){
    printf("One");
}else{ if(a == 2){
        printf("Two");
    }else{ if(a == 3){
            printf("Three");
        }else{ if(a == 4){
                printf("Four");}}}}
```



Hmm … so an if-else combination counts as a statement?

...rackets

...metimes, to make code look pretty, professional ...grammers omit brackets when not r...



Yup!

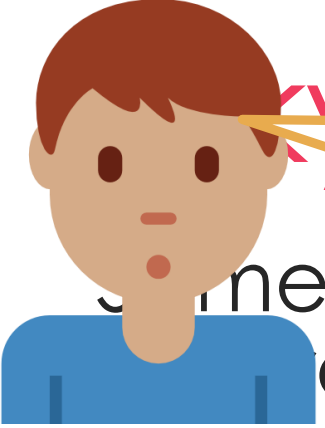
Better indentation

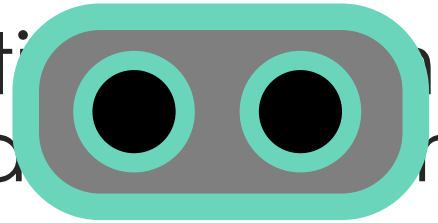
```
if(a == 1)
    printf("One");
else if(a == 2)
    printf("Two");
else if(a == 3)
    printf("Three");
else if(a == 4)
    printf("Four");
```

SAME OUTPUT

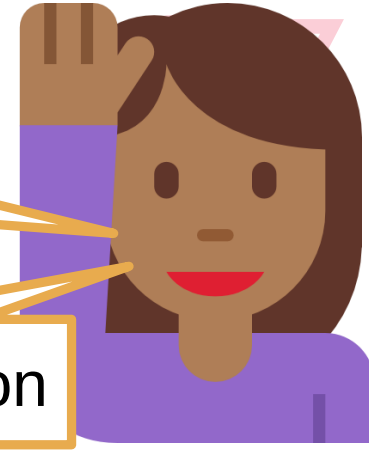
```
if(a == 1){
    printf("One");
}else{ if(a == 2){
        printf("Two");
    }else{ if(a == 3){
            printf("Three");
        }else{ if(a == 4){
                printf("Four");}}}}
```

rackets





ake code look pretty, professional
nit brackets when not r

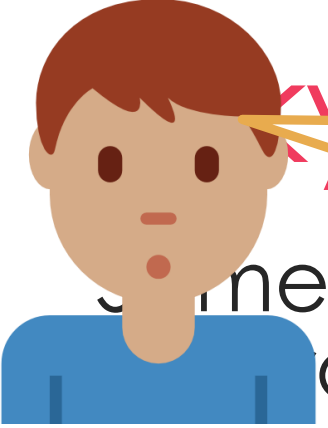



```
if(a == 1)
    printf("One");
else if(a == 2)
    printf("Two");
else if(a == 3)
    printf("Three");
else if(a == 4)
    printf("Four");
```

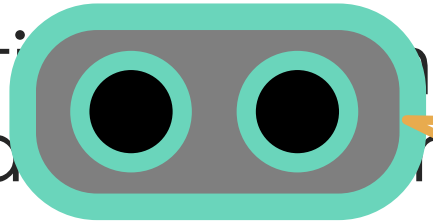
SAME OUTPUT

```
if(a == 1){
    printf("One");
}else{ if(a == 2){
        printf("Two");
    }else{ if(a == 3){
            printf("Three");
        }else{ if(a == 4){
                printf("Four");}}}}
```

rackets



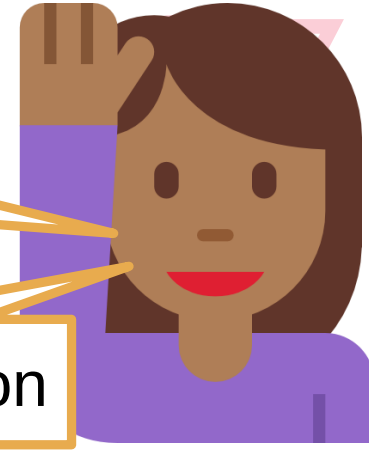
Hmm … so an if-else combination counts as a statement?



Remember, we discussed *nested* if-else statements

Yup!

Better indentation



```
if(a == 1)
    printf("One");
else if(a == 2)
    printf("Two");
else if(a == 3)
    printf("Three");
else if(a == 4)
    printf("Four");
```

SAME OUTPUT

```
if(a == 1){
    printf("One");
}else{ if(a == 2){
        printf("Two");
    }else{ if(a == 3){
            printf("Three");
        }else{ if(a == 4){
                printf("Four");}}}}
```

Unsafe Practices

Here as well, if you do not put curly brackets, Mr. C will try to put them for you

Unsafe Practices

Here as well, if you do not put curly brackets, Mr. C will try to put them for you

Unsafe Practices

Here as well, if you do not put curly brackets, Mr. C will try to put them for you

```
if((a != 0) && (b != 0))
```

Unsafe Practices

Here as well, if you do not put curly brackets, Mr. C will try to put them for you

```
if((a != 0) && (b != 0))
    if(a * b >= 0)
```

Unsafe Practices

Here as well, if you do not put curly brackets, Mr. C will try to put them for you

```
if((a != 0) && (b != 0))
    if(a * b >= 0)
        printf("Positive product");
```

Unsafe Practices

Here as well, if you do not put curly brackets, Mr. C will try to put them for you

```
if((a != 0) && (b != 0))
    if(a * b >= 0)
        printf("Positive product");
else
```

Unsafe Practices

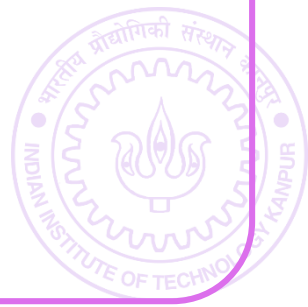
Here as well, if you do not put curly brackets, Mr. C will try to put them for you

```
if((a != 0) && (b != 0))
    if(a * b >= 0)
        printf("Positive product");
else
    printf("One number is zero");
```

Unsafe Practices

Here as well, if you do not put curly brackets, Mr. C will try to put them for you

```
if((a != 0) && (b != 0))
    if(a * b >= 0)
        printf("Positive product");
else
    printf("One number is zero");
```

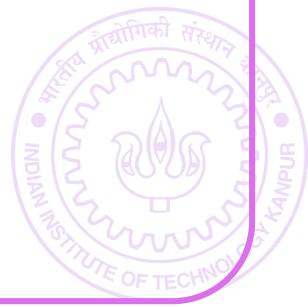


Unsafe Practices

Here as well, if you do not put curly brackets, Mr. C will try to put them for you

```
if((a != 0) && (b != 0))
    if(a * b >= 0)
        printf("Positive product");
else
    printf("One number is zero");
```

```
if((a != 0) && (b != 0)){
```

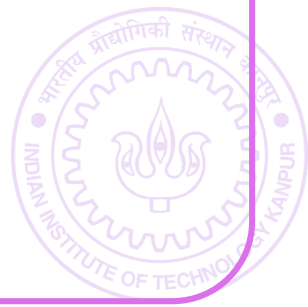


Unsafe Practices

Here as well, if you do not put curly brackets, Mr. C will try to put them for you

```
if((a != 0) && (b != 0))
    if(a * b >= 0)
        printf("Positive product");
else
    printf("One number is zero");
```

```
if((a != 0) && (b != 0)){
    if(a * b >= 0){
```

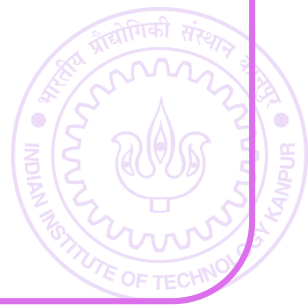


Unsafe Practices

Here as well, if you do not put curly brackets, Mr. C will try to put them for you

```
if((a != 0) && (b != 0))
    if(a * b >= 0)
        printf("Positive product");
else
    printf("One number is zero");
```

```
if((a != 0) && (b != 0)){
    if(a * b >= 0){
        printf("Positive product");
```

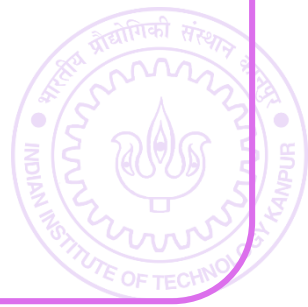


Unsafe Practices

Here as well, if you do not put curly brackets, Mr. C will try to put them for you

```
if((a != 0) && (b != 0))
    if(a * b >= 0)
        printf("Positive product");
else
    printf("One number is zero");
```

```
if((a != 0) && (b != 0)){
    if(a * b >= 0){
        printf("Positive product");
    }else{
```



Unsafe Practices

Here as well, if you do not put curly brackets, Mr. C will try to put them for you

```
if((a != 0) && (b != 0))
    if(a * b >= 0)
        printf("Positive product");
else
    printf("One number is zero");
```

```
if((a != 0) && (b != 0)){
    if(a * b >= 0){
        printf("Positive product");
    }else{
        printf("One number is zero");
```

Unsafe Practices

Here as well, if you do not put curly brackets, Mr. C will try to put them for you

```
if((a != 0) && (b != 0))
    if(a * b >= 0)
        printf("Positive product");
else
    printf("One number is zero");
```

```
if((a != 0) && (b != 0)){
    if(a * b >= 0){
        printf("Positive product");
    }else{
        printf("One number is zero");
    }
```

Unsafe Practices

Here as well, if you do not put curly brackets, Mr. C will try to put them for you

```
if((a != 0) && (b != 0))
    if(a * b >= 0)
        printf("Positive product");
else
    printf("One number is zero");
```

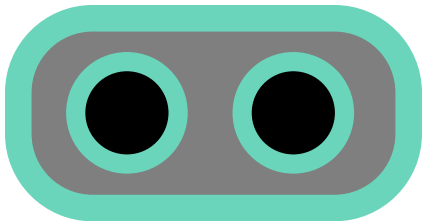
```
if((a != 0) && (b != 0)){
    if(a * b >= 0){
        printf("Positive product");
    }else{
        printf("One number is zero");
    }
}
```

Unsafe Practices

Here as well, if you do not put curly brackets, Mr. C will try to put them for you

```
if((a != 0) && (b != 0))
    if(a * b >= 0)
        printf("Positive product");
else
    printf("One number is zero");
```

```
if((a != 0) && (b != 0)){
    if(a * b >= 0){
        printf("Positive product");
    }else{
        printf("One number is zero");
    }
}
```

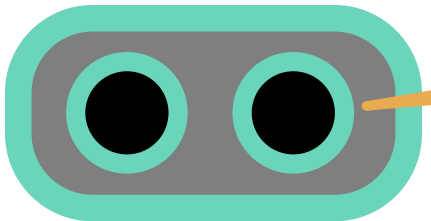


Unsafe Practices

Here as well, if you do not put curly brackets, Mr. C will try to put them for you

```
if((a != 0) && (b != 0))
    if(a * b >= 0)
        printf("Positive product");
else
    printf("One number is zero");
```

```
if((a != 0) && (b != 0)){
    if(a * b >= 0){
        printf("Positive product");
    }else{
        printf("One number is zero");
    }
}
```



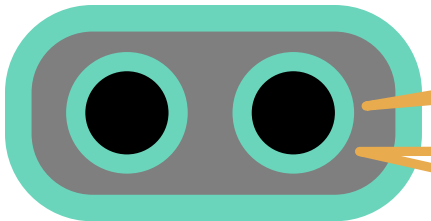
If you do not put brackets, I will match else to closest if

Unsafe Practices

Here as well, if you do not put curly brackets, Mr. C will try to put them for you

```
if((a != 0) && (b != 0))
    if(a * b >= 0)
        printf("Positive product");
else
    printf("One number is zero");
```

```
if((a != 0) && (b != 0)){
    if(a * b >= 0){
        printf("Positive product");
    }else{
        printf("One number is zero");
    }
}
```



If you do not put brackets, I will match else to closest if

I will not care how you did indentation

Unsafe Practices

Unsafe Practices

You may have an if without an else, but you cannot have an else without an if

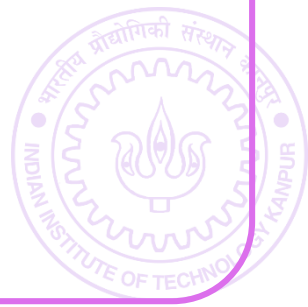
Unsafe Practices

You may have an if without an else, but you cannot have an else without an if

Unsafe Practices

You may have an if without an else, but you cannot have an else without an if

```
if(a == 5)
```



Unsafe Practices

You may have an if without an else, but you cannot have an else without an if

```
if(a == 5)
    printf("Five");
```

Unsafe Practices

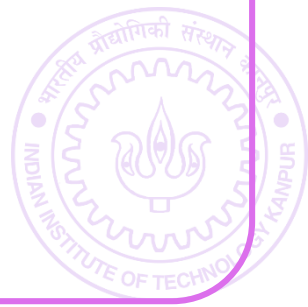
You may have an if without an else, but you cannot have an else without an if

```
if(a == 5)
    printf("Five");
else
```

Unsafe Practices

You may have an if without an else, but you cannot have an else without an if

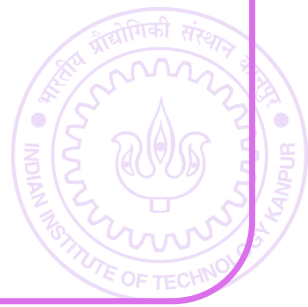
```
if(a == 5)
    printf("Five");
else
    printf("Not Five");
```



Unsafe Practices

You may have an if without an else, but you cannot have an else without an if

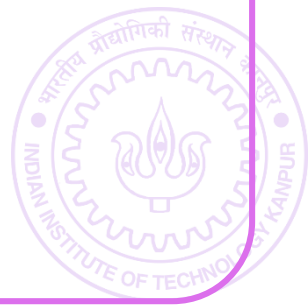
```
if(a == 5)
    printf("Five");
else
    printf("Not Five");
else
```



Unsafe Practices

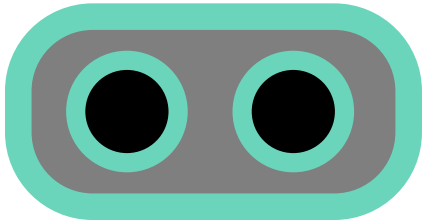
You may have an if without an else, but you cannot have an else without an if

```
if(a == 5)
    printf("Five");
else
    printf("Not Five");
else
    printf("Something Else");
```



Unsafe Practices

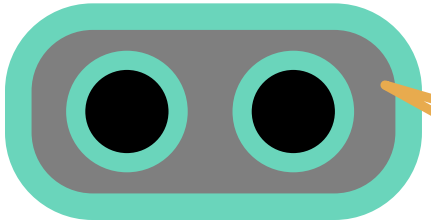
You may have an if without an else, but you cannot have an else without an if



```
if(a == 5)
    printf("Five");
else
    printf("Not Five");
else
    printf("Something Else");
```

Unsafe Practices

You may have an if without an else, but you cannot have an else without an if

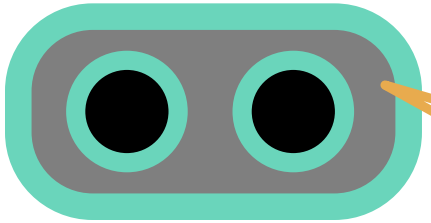


Does not make sense to me

```
if(a == 5)
    printf("Five");
else
    printf("Not Five");
else
    printf("Something Else");
```

Unsafe Practices

You may have an if without an else, but you cannot have an else without an if

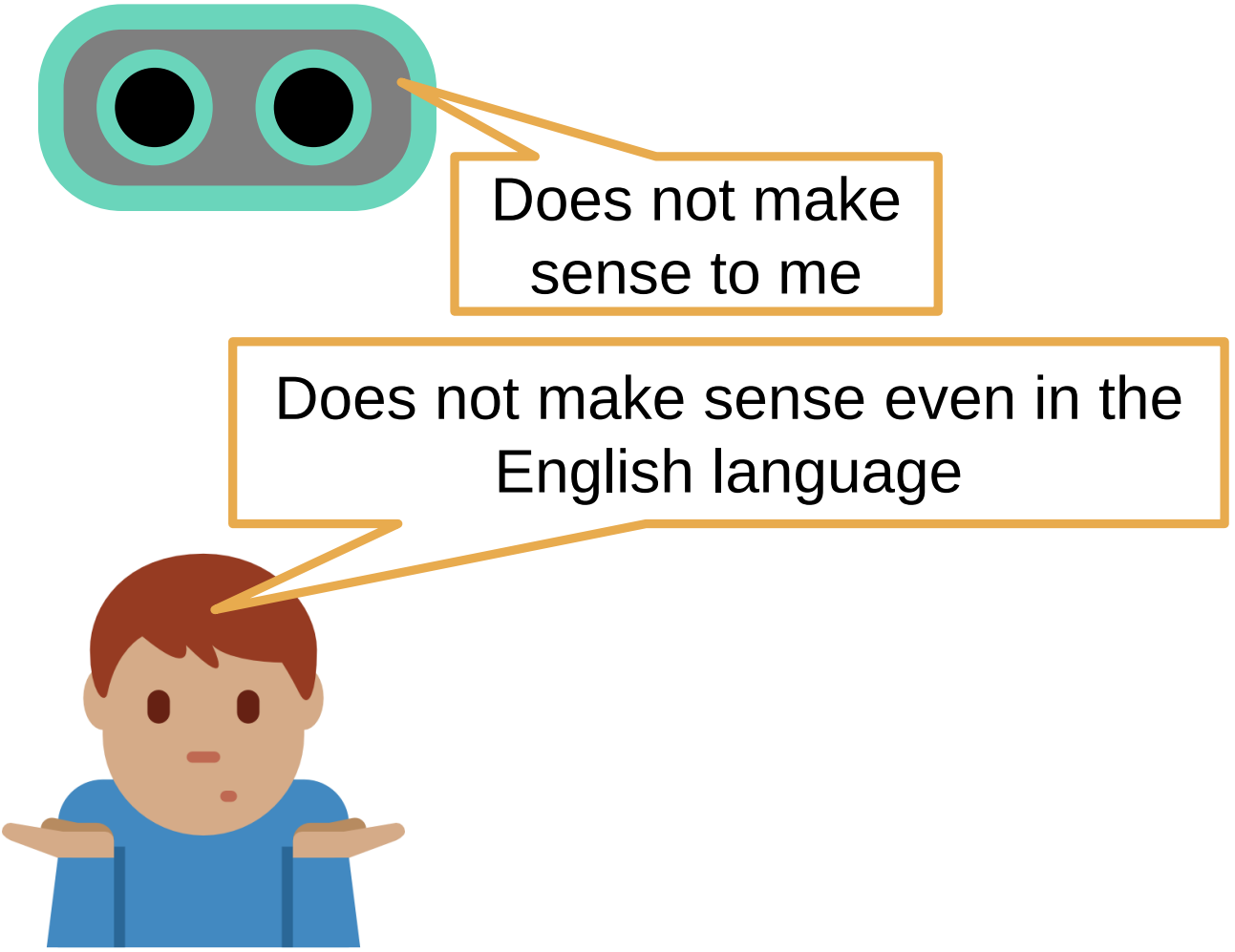


Does not make sense to me

```
if(a == 5)
    printf("Five");
else
    printf("Not Five");
else
    printf("Something Else");
```

Unsafe Practices

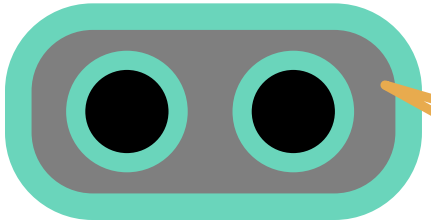
You may have an if without an else, but you cannot have an else without an if




```
if(a == 5)
    printf("Five");
else
    printf("Not Five");
else
    printf("Something Else");
```

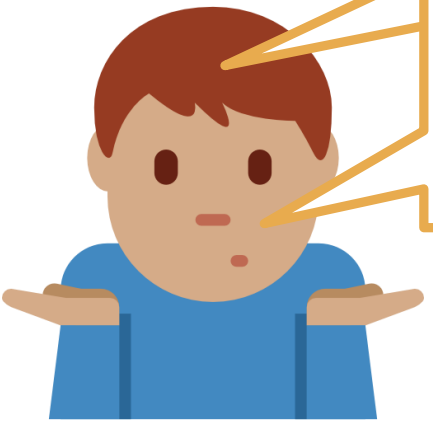
Unsafe Practices

You may have an if without an else, but you cannot have an else without an if



Does not make sense to me

Does not make sense even in the English language



If you are hungry go to the mess, else go to the lab, else go to the ShopC

```
if(a == 5)
    printf("Five");
else
    printf("Not Five");
else
    printf("Something Else");
```