Mr. C doesn’t get bored

ESC101: Fundamentals of Computing

Purushottam Kar

Announcements - Quiz

- Major quiz **this** week – (syllabus till **Friday Aug 24**)
  - Wednesday, August 29, 2018, 12PM-12:50PM, L20 (i.e. lecture hour)
  - During lecture hours – don't be absent
  - **Bring your institute ID card** with you – will lose time if you forget
  - No minor quizzes during lab this week (August 27-August 30)
- Bring a **pencil, eraser and sharpener** with you
  - Answers to be written on question paper itself and returned back
  - If you make a mistake with pen – no extra question papers
  - If unsure, **first write answer with pencil** and **finally write it in pen**
  - We WONT HAVE EXTRA QUESTION PAPERS in case you spoil yours
  - We WONT HAVE PENCILS, ERASERS in case you forget

Announcements - Grade

- Minor quiz for week 2 has been graded
  - Marks have been uploaded to Gradescope
  - You would have received an email inviting you to join Gradescope
  - In case of issues, please ask us during your lab this week
- Minor quiz has been autograded
  - No "grace" marks – negative penalty for useless regrading requests
  - Yesterday, someone awarded -2 marks for such a regrading request
- If you cannot see your minor quiz on Gradescope
  - You gave your minor quiz on pen-paper – please wait for a day
  - You gave the wrong secret code (someone wrote their mobile no ☺)
  - Contact us over Piazza (private message) if you feel something wrong

Announcements - Holiday

- Institute holiday next Monday (03 September, 2018)
- No lecture, no lab on that day
- Extra lecture on Saturday 08 September, 2018
  - 12 noon, L20 (same as usual)
  - Scheduled by DoAA, not by me – I like to sleep on Sat too ☹
- Extra lab for B1, B2, B3 on Saturday 08 September
  - 2PM – 5PM, New Core Labs CC-02 (same as usual)

Announcements - Exam

- Mid-sem lab exam on 09 September, 2018 (Sunday)
- Will give details of the same very soon
- Please do not go on holiday or travel – no make up!

Humans vs Mr. C

Humans vs Mr. C

- Humans get bored and tired easily

Humans vs Mr. C

- Humans get bored and tired easily
  - Especially when a similar job has to be repeated again and again

Humans vs Mr. C

- Humans get bored and tired easily
  - Especially when a similar job has to be repeated again and again
  - E.g. go through all minor quiz submissions and check where all have students entered mobile phone numbers instead of secret code

Humans vs Mr. C

- Humans get bored and tired easily
  - Especially when a similar job has to be repeated again and again
  - E.g. go through all minor quiz submissions and check where all have students entered mobile phone numbers instead of secret code
  - E.g. look at all pages on the internet and select those which are important for the ESC101 course (Google has to do this)

Humans vs Mr. C

- Humans get bored and tired easily
  - Especially when a similar job has to be repeated again and again
  - E.g. go through all minor quiz submissions and check where all have students entered mobile phone numbers instead of secret code
  - E.g. look at all pages on the internet and select those which are important for the ESC101 course (Google has to do this)
- Mr. C is much stronger and durable ☺

Humans vs Mr. C

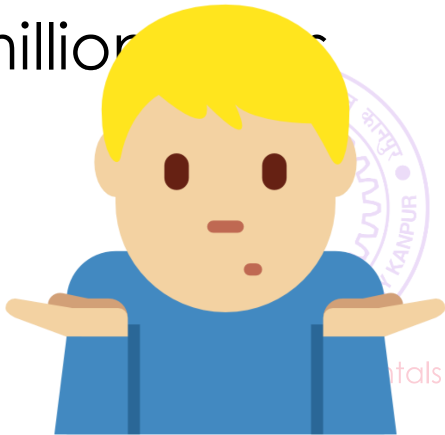
- Humans get bored and tired easily
  - Especially when a similar job has to be repeated again and again
  - E.g. go through all minor quiz submissions and check where all have students entered mobile phone numbers instead of secret code
  - E.g. look at all pages on the internet and select those which are important for the ESC101 course (Google has to do this)
- Mr. C is much stronger and durable ☺
  - Does not get bored even if we ask similar job to be done million times

Humans vs Mr. C

- Humans get bored and tired easily
  - Especially when a similar job has to be repeated again and again
  - E.g. go through all minor quiz submissions and check where all have students entered mobile phone numbers instead of secret code
  - E.g. look at all pages on the internet and select those which are important for the ESC101 course (Google has to do this)
- Mr. C is much stronger and durable ☺
  - Does not get bored even if we ask similar job to be done million times
  - Does not make mistakes after getting tired

Humans vs Mr. C

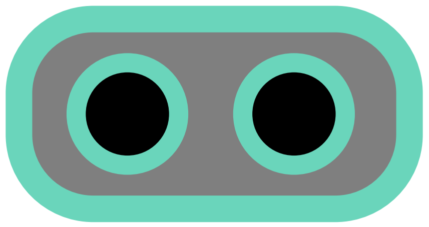
- Humans get bored and tired easily
  - Especially when a similar job has to be repeated again and again
  - E.g. go through all minor quiz submissions and check where all have students entered mobile phone numbers instead of secret code
  - E.g. look at all pages on the internet and select those which are important for the ESC101 course (Google has to do this)
- Mr. C is much stronger and durable ☺
  - Does not get bored even if we ask similar job to be done million
  - Does not make mistakes after getting tired



Humans vs Mr. C

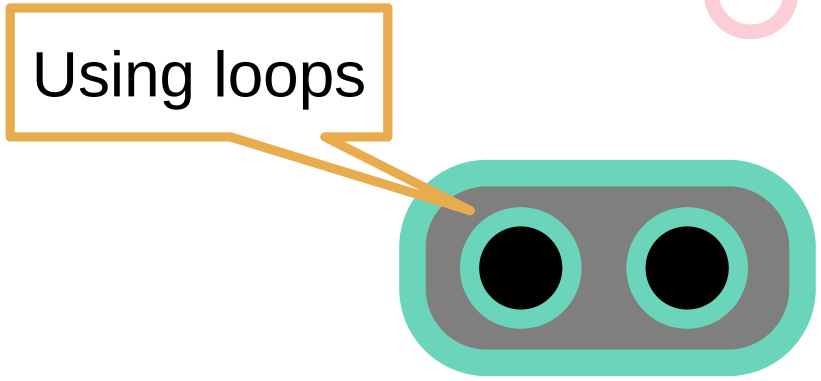
- Humans get bored and tired easily
  - Especially when a similar job has to be repeated again and again
  - E.g. go through all minor quiz submissions and check where all have students entered mobile phone numbers instead of secret code
  - E.g. look at all pages on the internet and select those which are important for the ESC101 course (Google has to do this)
- Mr. C is much stronger and durable ☺
  - Does not get bored even if we ask similar job to be done million
  - Does not make mistakes after getting ti

Humans vs Mr. C



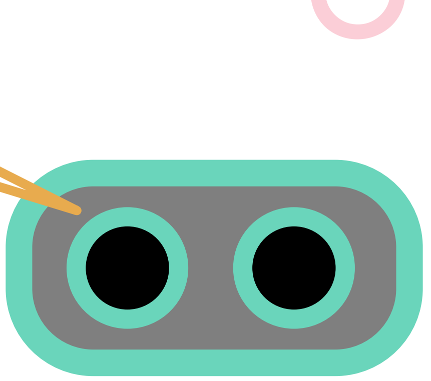
- Humans get bored and tired easily
  - Especially when a similar job has to be repeated again and again
  - E.g. go through all minor quiz submissions and check where all have students entered mobile phone numbers instead of secret code
  - E.g. look at all pages on the internet and select those which are important for the ESC101 course (Google has to do this)
- Mr. C is much stronger and durable ☺
  - Does not get bored even if we ask similar job to be done million
  - Does not make mistakes after getting ti

Humans vs Mr. C



- Humans get bored and tired easily
  - Especially when a similar job has to be repeated again and again
  - E.g. go through all minor quiz submissions and check where all have students entered mobile phone numbers instead of secret code
  - E.g. look at all pages on the internet and select those which are important for the ESC101 course (Google has to do this)
- Mr. C is much stronger and durable ☺
  - Does not get bored even if we ask similar job to be done million
  - Does not make mistakes after getting ti

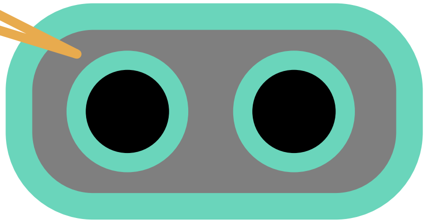
Humans vs Mr. C

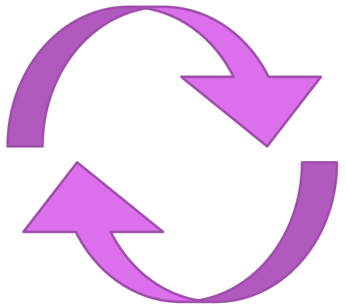


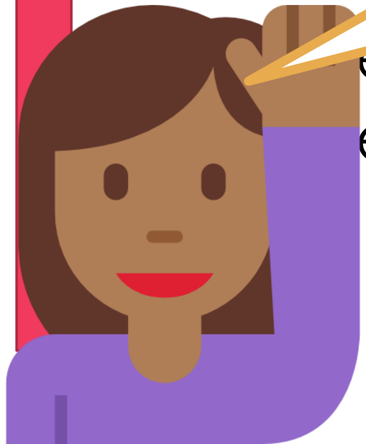
- Humans get bored and tired easily
  - Especially when a similar job has to be repeated again and again
  - E.g. go through all minor quiz submissions and check where all have students entered mobile phone numbers instead of secret code
  - E.g. look at all pages on the internet and select those which are important for the ESC101 course (Google has to do this)
- Mr. C is much stronger and durable ☺
  - es not get bored even if we ask similar job to be done million
  - es not make mistakes after getting ti

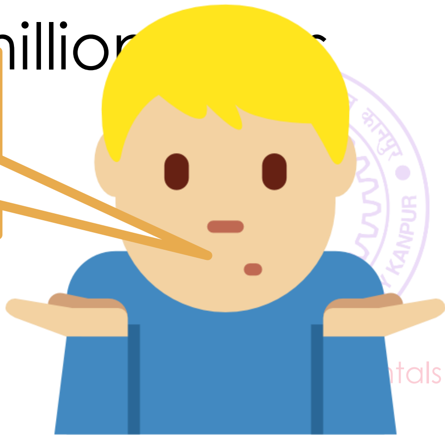
Humans vs Mr. C

- Humans get bored and tired easily
  - Especially when a similar job has to be repeated again and again
  - E ssions and check where all have s nbers instead of secret code
  - E t and select those which are i oogle has to do this)
- Mr. C durable ☺
  - es not get bored even if we ask similar job to be done millior
  - es not make mistakes after getting ti



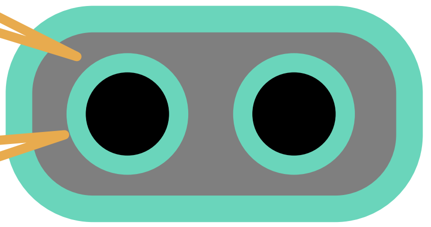

The English word “loop” means something that goes round and round. Is there a connection?

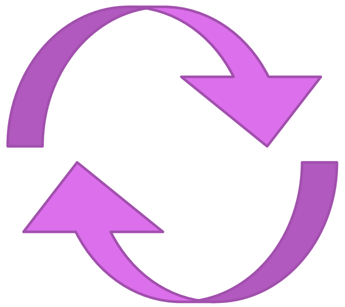




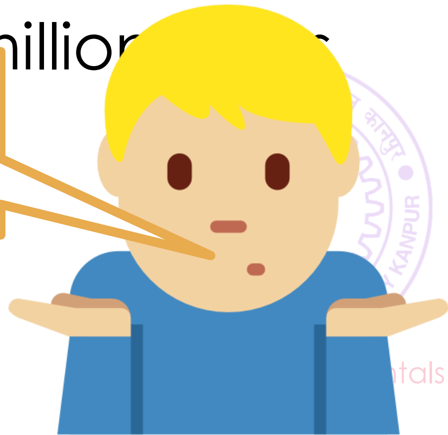
Humans vs Mr. C

- Humans get bore
  - Especially when a similar job has to be repeated again and again
  - E ssions and check where all have s nbers instead of secret code
  - E t and select those which are i oogle has to do this)
- Mr. C en stronger and durable ☺
  - es not get bored even if we ask similar job to be done million
  - es not make mistakes after getting ti


Using loops
Yes. I keep executing a set of tasks that you give me again and again till you ask me to stop

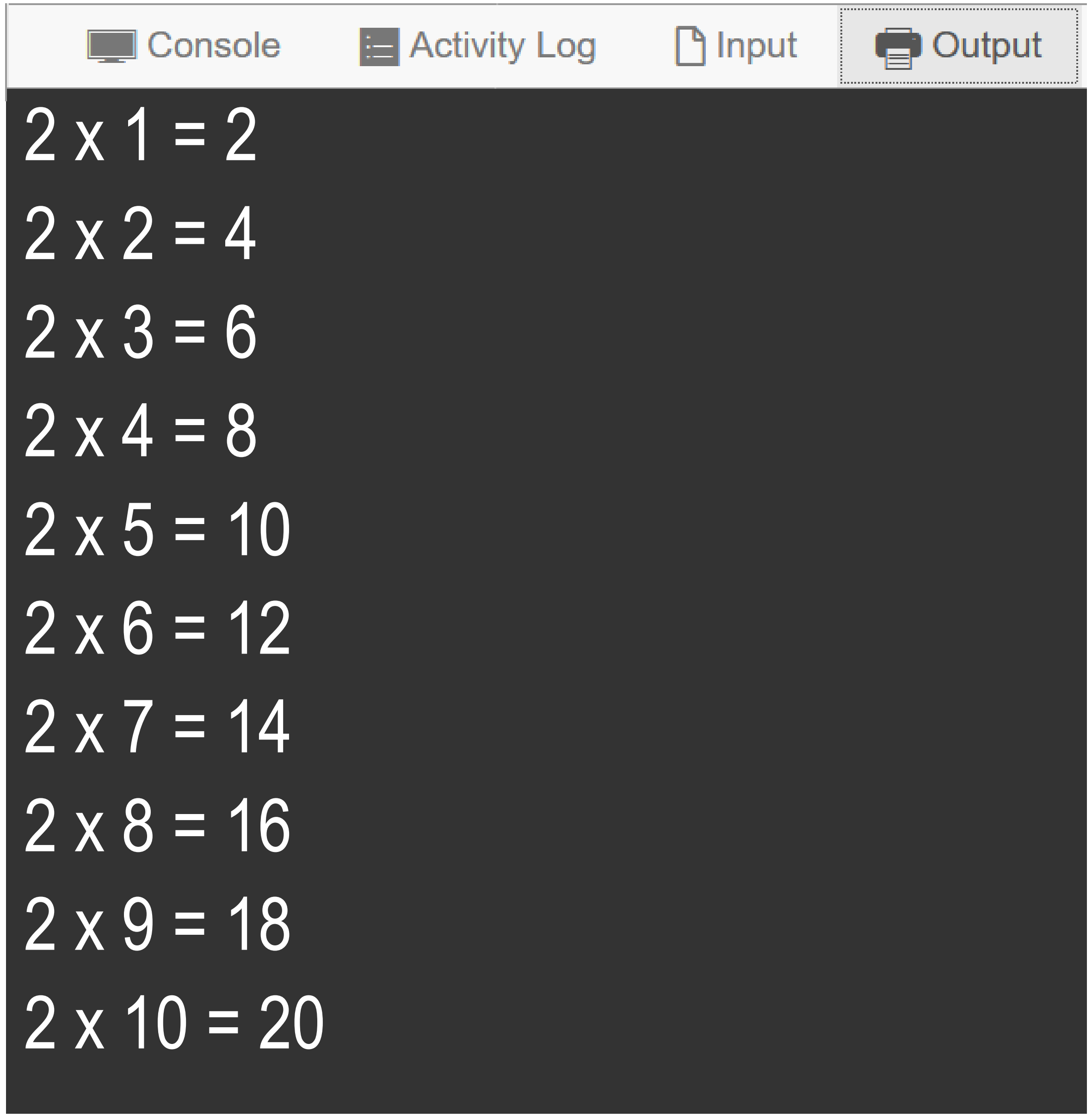


The English word "loop" means something that goes round and round. Is there a connection?



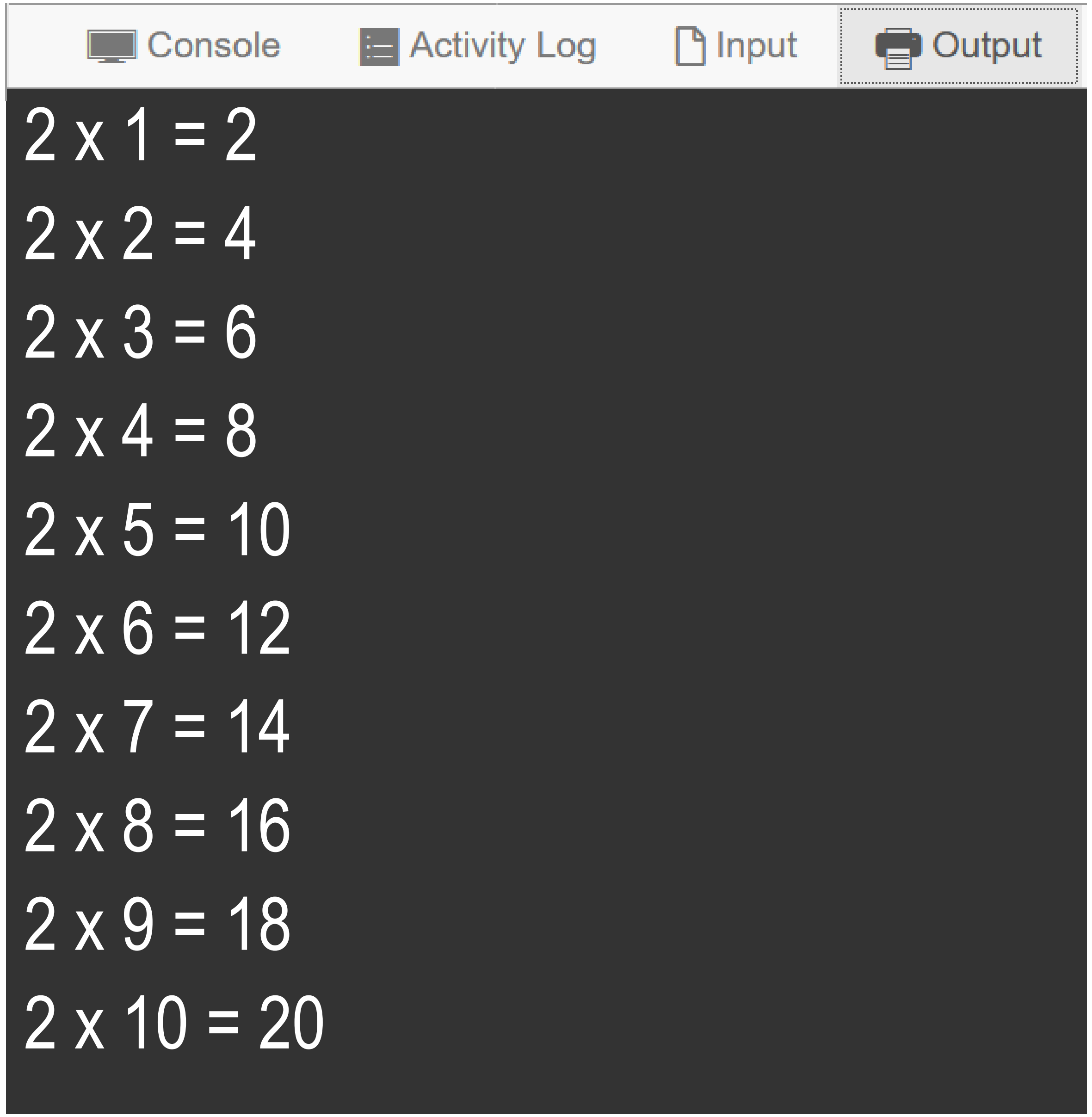
Printing the multiplication table of 2

Printing the multiplication table of 2



```
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18
2 x 10 = 20
```

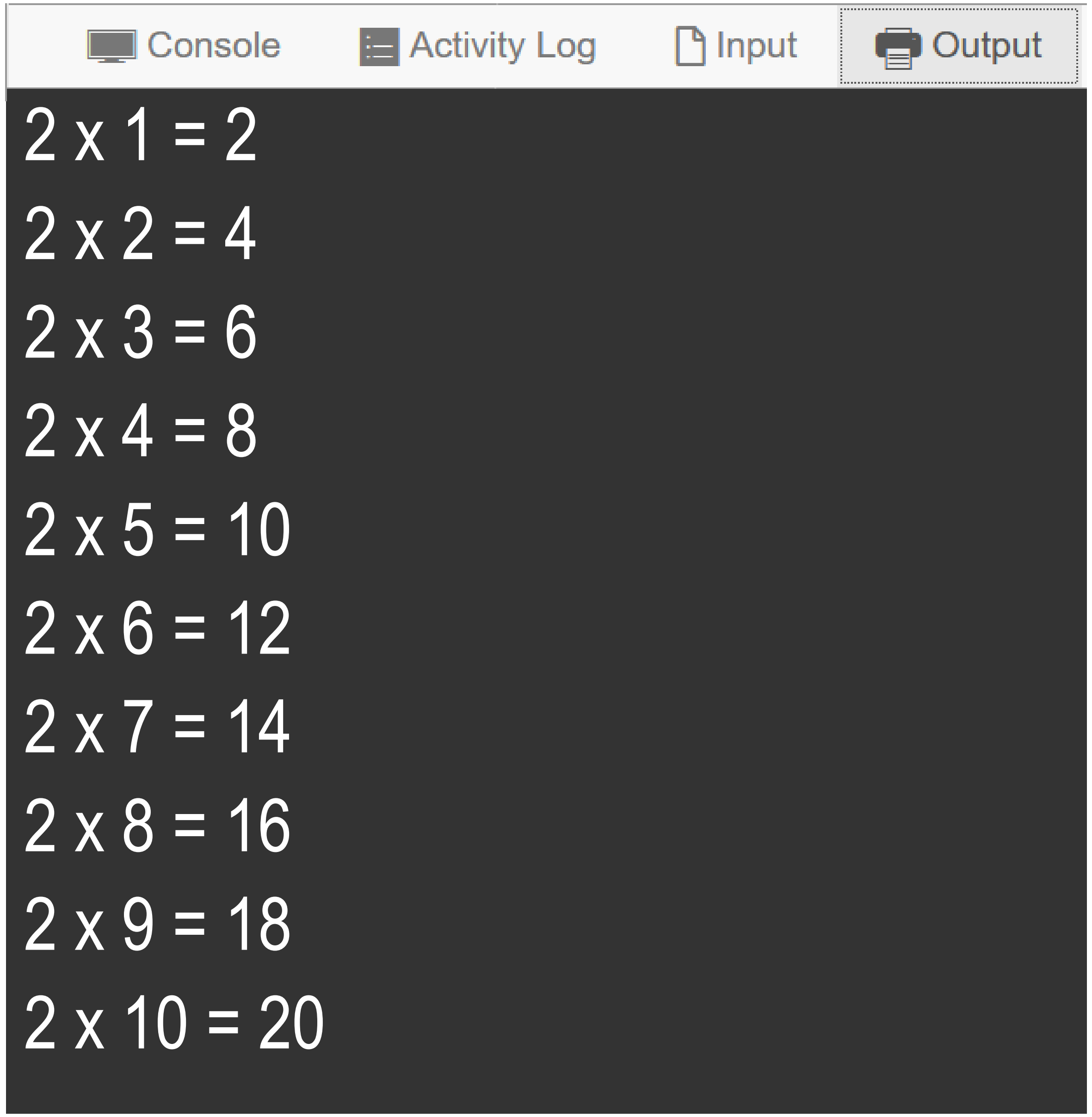
Printing the multiplication table of 2



```
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18
2 x 10 = 20
```

```
printf("2 x 1 = 2\n");
printf("2 x 2 = 4\n");
printf("2 x 3 = 6\n");
printf("2 x 4 = 8\n");
printf("2 x 5 = 10\n");
printf("2 x 6 = 12\n");
printf("2 x 7 = 14\n");
printf("2 x 8 = 16\n");
printf("2 x 9 = 18\n");
printf("2 x 10 = 20\n");
```

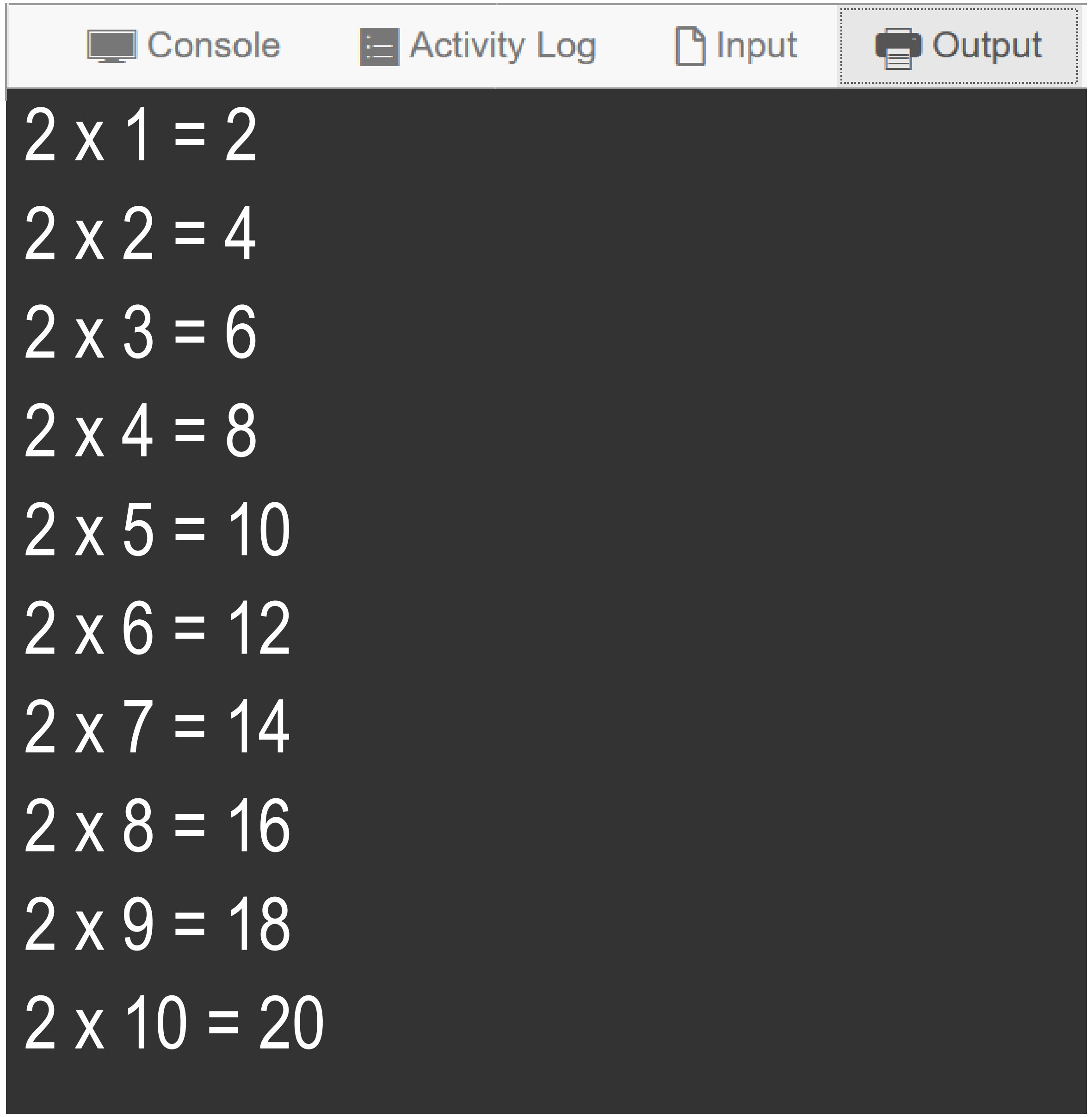
Printing the multiplication table of 2



```
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18
2 x 10 = 20
```

```
int a = 2, b = 1;
printf("%d x %d = %d\n", a, b, a*b);
b++;
printf("%d x %d = %d\n", a, b, a*b);
b++;
printf("%d x %d = %d\n", a, b, a*b);
b++;
printf("%d x %d = %d\n", a, b, a*b);
b++;
…
```

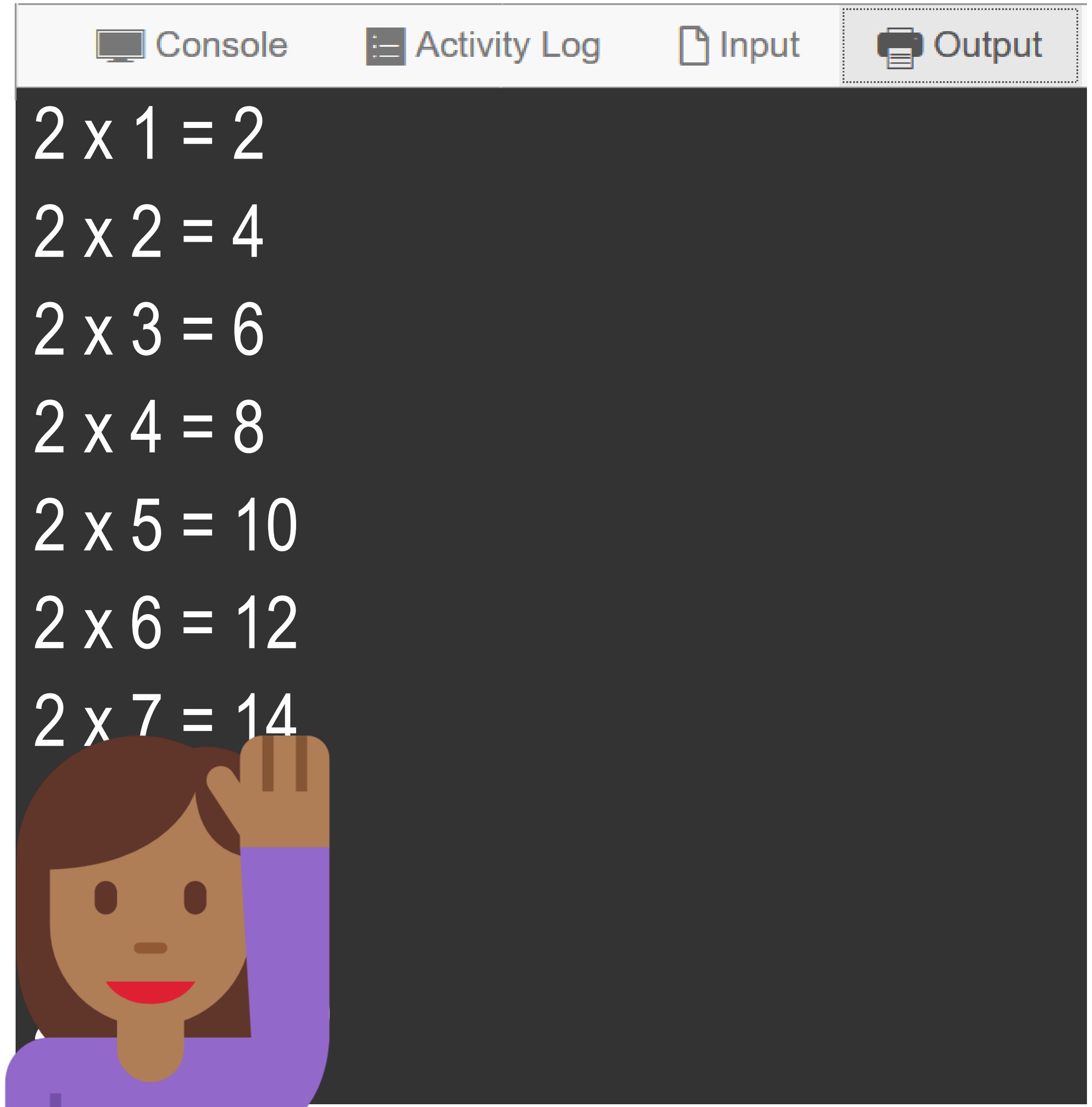
Printing the multiplication table of 2



```
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18
2 x 10 = 20
```

```
printf("2 x 1 = 2\n");
printf("2 x 2 = 4\n");
printf("2 x 3 = 6\n");
printf("2 x 4 = 8\n");
printf("2 x 5 = 10\n");
printf("2 x 6 = 12\n");
printf("2 x 7 = 14\n");
printf("2 x 8 = 16\n");
printf("2 x 9 = 18\n");
printf("2 x 10 = 20\n");
```

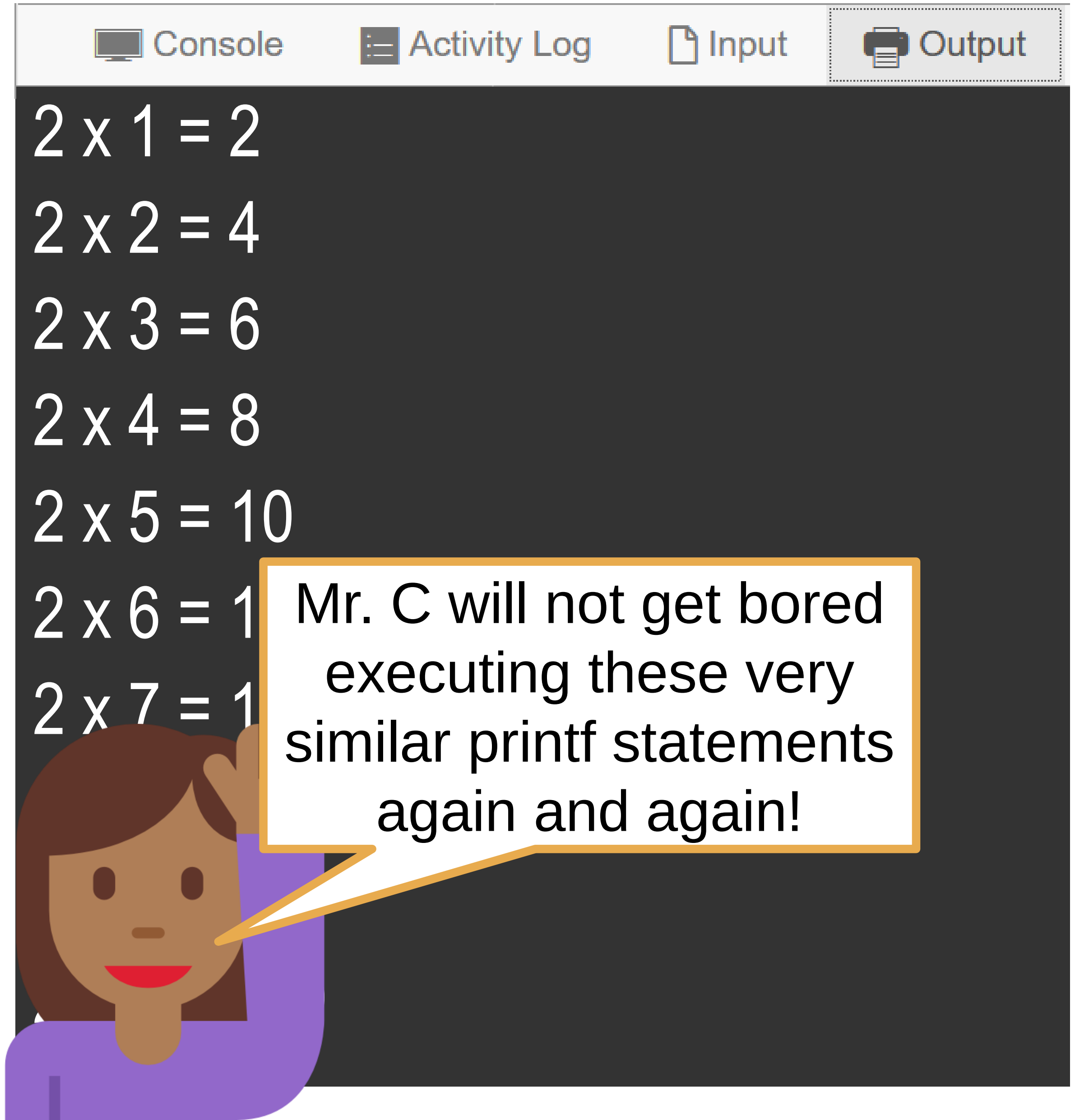
Printing the multiplication table of 2



```
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
```

```
printf("2 x 1 = 2\n");
printf("2 x 2 = 4\n");
printf("2 x 3 = 6\n");
printf("2 x 4 = 8\n");
printf("2 x 5 = 10\n");
printf("2 x 6 = 12\n");
printf("2 x 7 = 14\n");
printf("2 x 8 = 16\n");
printf("2 x 9 = 18\n");
printf("2 x 10 = 20\n");
```

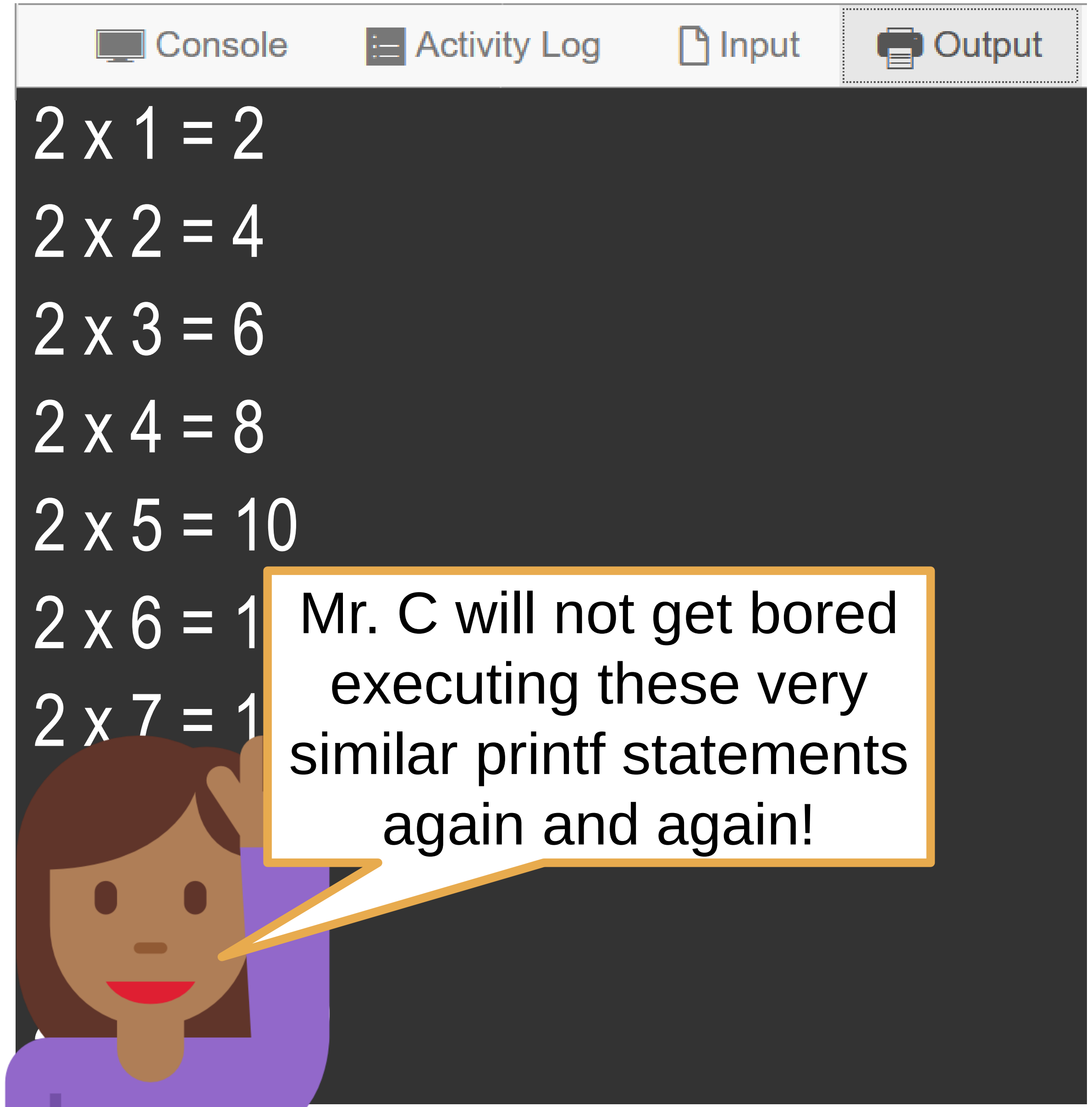
Printing the multiplication table of 2


Console
Activity Log
Input
Output
Mr. C will not get bored executing these very similar printf statements again and again!

```
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 1
2 x 7 = 1
```

```
printf("2 x 1 = 2\n");
printf("2 x 2 = 4\n");
printf("2 x 3 = 6\n");
printf("2 x 4 = 8\n");
printf("2 x 5 = 10\n");
printf("2 x 6 = 12\n");
printf("2 x 7 = 14\n");
printf("2 x 8 = 16\n");
printf("2 x 9 = 18\n");
printf("2 x 10 = 20\n");
```

Printing the multiplication table of 2

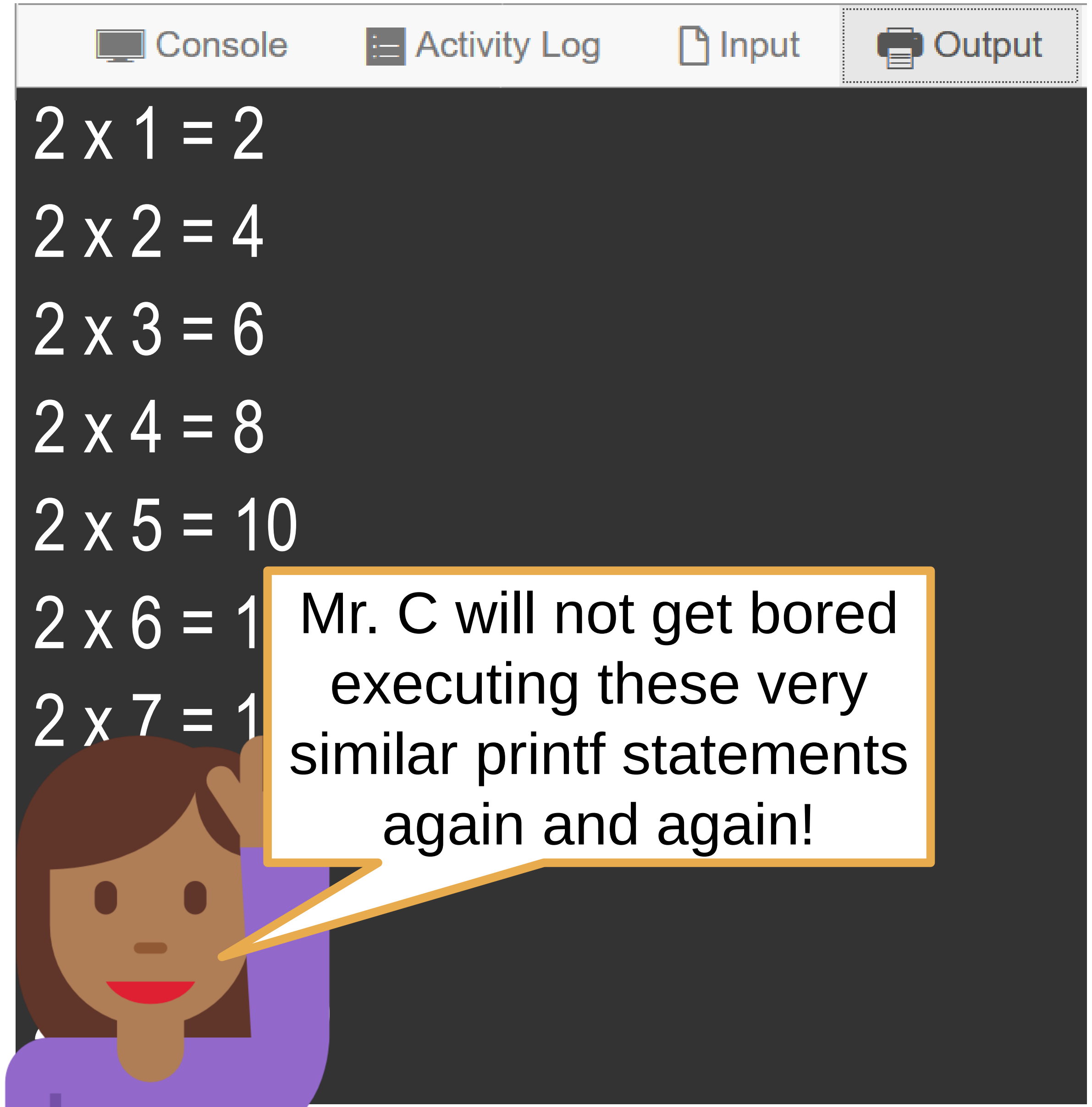


```
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 1
2 x 7 = 1
```

Mr. C will not get bored executing these very similar printf statements again and again!

```
printf("2 x 1 = 2\n");
printf("2 x 2 = 4\n");
printf("2 x 3 = 6\n");
printf("2 x 4 = 8\n");
printf("2 x 5 = 10\n");
printf("2 x 6 = 12\n");
printf("2 x 7 = 14\n");
printf("2 x 8 = 16\n");
printf("2 x 9 = 18\n");
printf("2 x 10 = 20\n");
```

Printing the multiplication table of 2

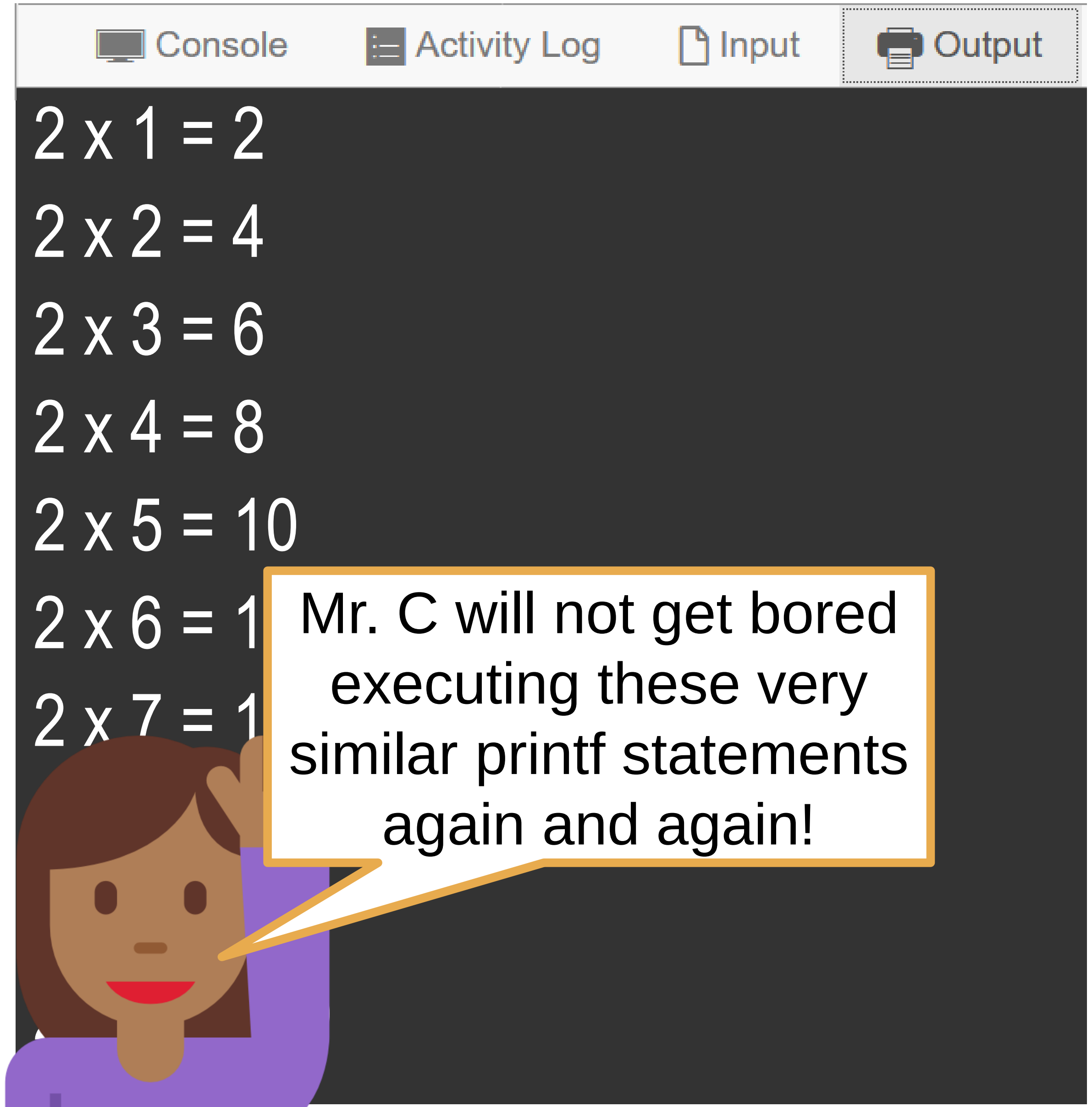


```
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 1
2 x 7 = 1
```

Mr. C will not get bored executing these very similar printf statements again and again!

```
printf(“2 x 1 = 2\n”);
printf(“2 x 2 = 4\n”);
printf(“2 x 3 = 6\n”);
printf(“2 x 4 = 8\n”);
printf(“2 x 5 = 10\n”);
printf(“2 x 6 = 12\n”);
printf(“2 x 7 = 14\n”);
printf(“2 x 8 = 16\n”);
printf(“2 x 9 = 18\n”);
printf(“
```

Printing the multiplication table of 2

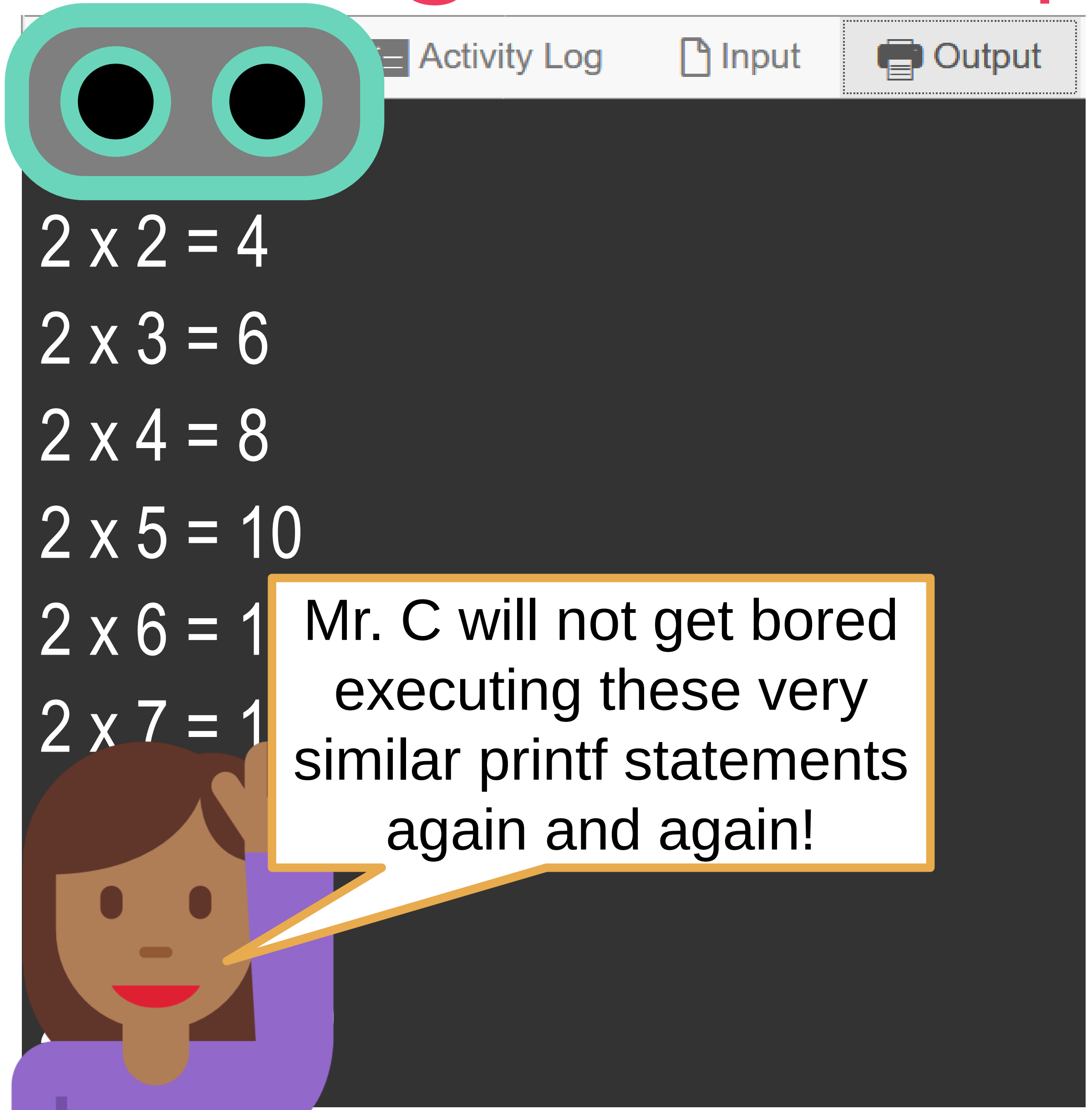


```
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 1
2 x 7 = 1
```

```
int a = 2, b = 1;
printf("%d x %d = %d\n", a, b, a*b);
b++;
printf("%d x %d = %d\n", a, b, a*b);
b++;
printf("%d x %d = %d\n", a, b, a*b);
b++;
printf("%d x %d = %d\n", a, b, a*b
b++;
...
```

Mr. C will not get bored executing these very similar printf statements again and again!

Printing the multiplication table of 2



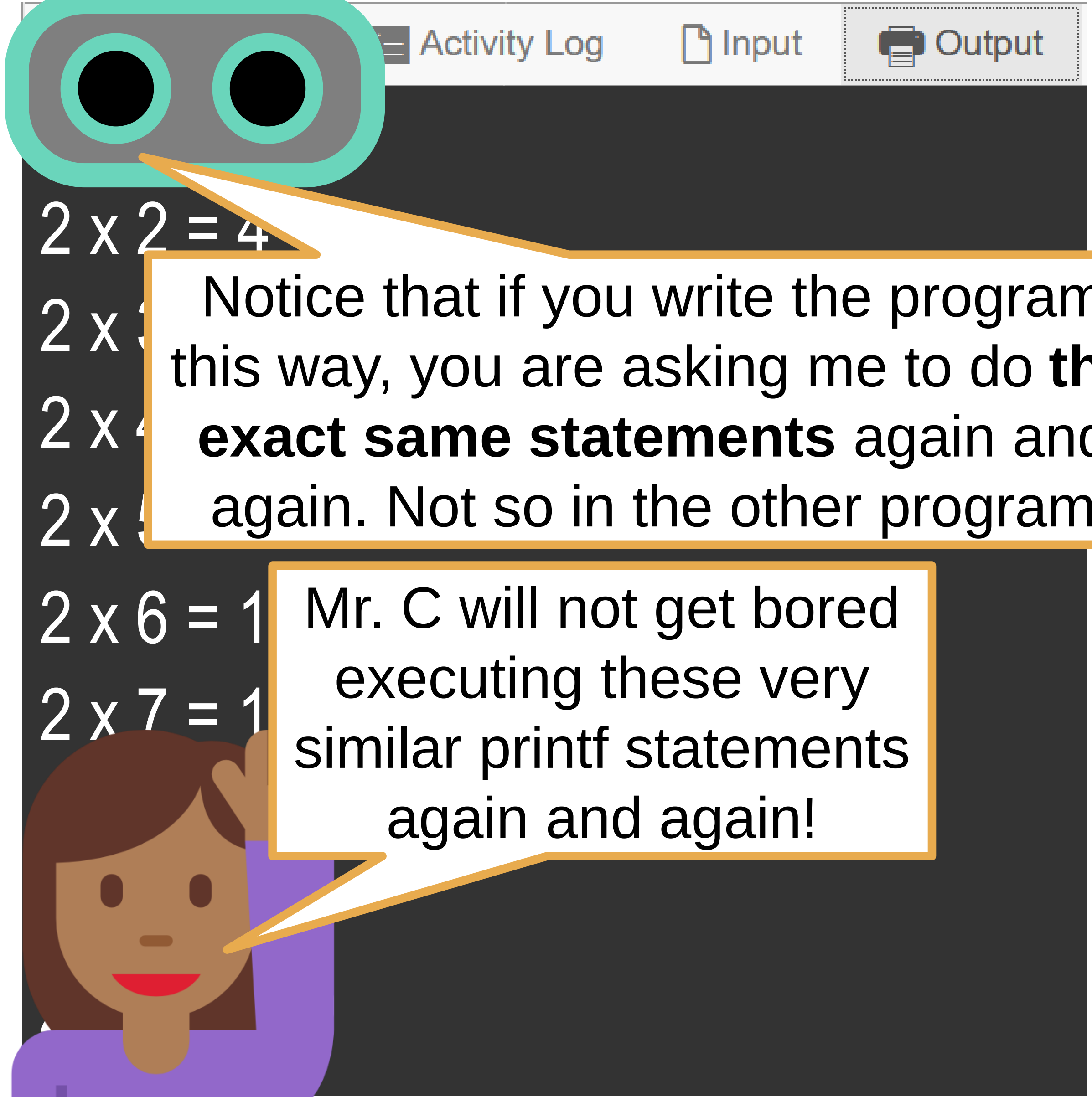
```
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 1
2 x 7 = 1
```

Mr. C will not get bored executing these very similar printf statements again and again!

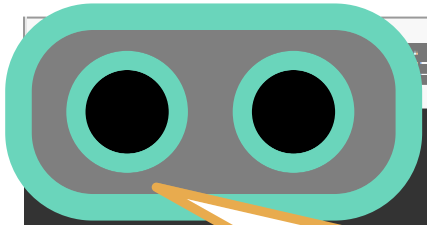
```
int a = 2, b = 1;
printf("%d x %d = %d\n", a, b, a*b);
b++;
printf("%d x %d = %d\n", a, b, a*b);
b++;
printf("%d x %d = %d\n", a, b, a*b);
b++;
printf("%d x %d = %d\n", a, b, a*b
b++;
…
```

Yes, but I got bored *writing* them ☹

Printing the multiplication table of 2



```
2 x 2 = 4
2 x 3
2 x 4
2 x 5
2 x 6 = 1
2 x 7 = 1
```



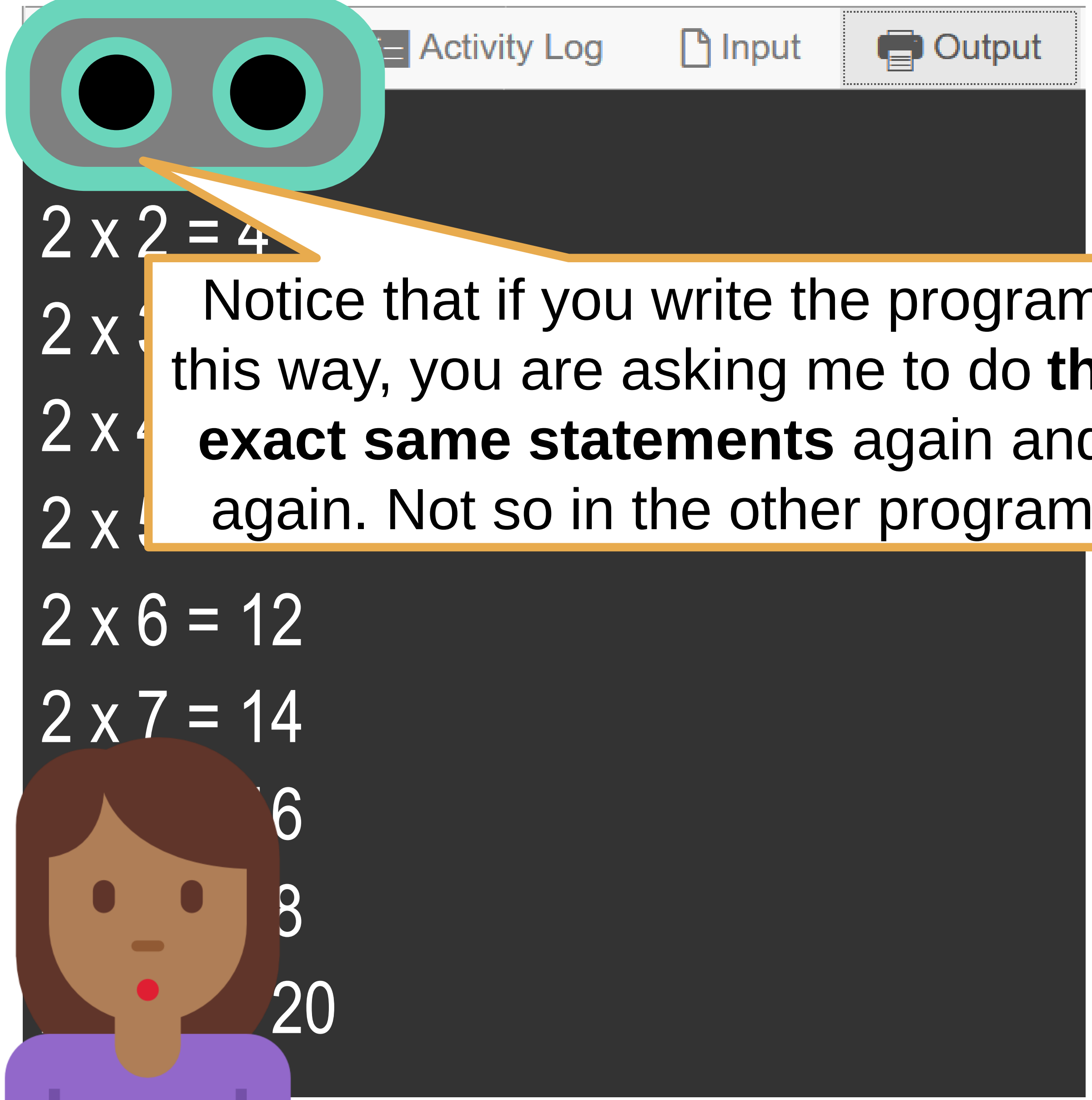
Notice that if you write the program this way, you are asking me to do **the exact same statements** again and again. Not so in the other program

Mr. C will not get bored executing these very similar printf statements again and again!

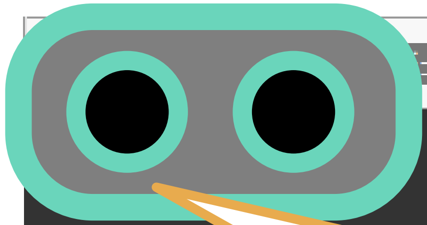
```
int a = 2, b = 1;
printf("%d x %d = %d\n", a, b, a*b);
b++;
printf("%d x %d = %d\n", a, b, a*b);
b++;
printf("%d x %d = %d\n", a, b, a*b);
b++;
printf("%d x %d = %d\n", a, b, a*b
b++;
...
```

Yes, but I got bored *writing* them ☹

Printing the multiplication table of 2



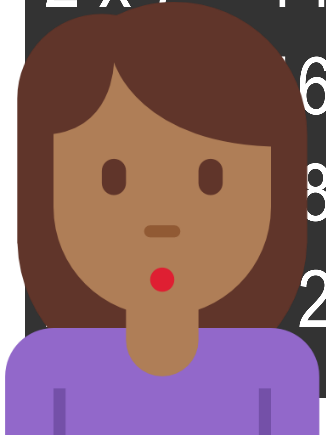
```
2 x 2 = 4
2 x
2 x
2 x
2 x 6 = 12
2 x 7 = 14
        6
        8
        20
```



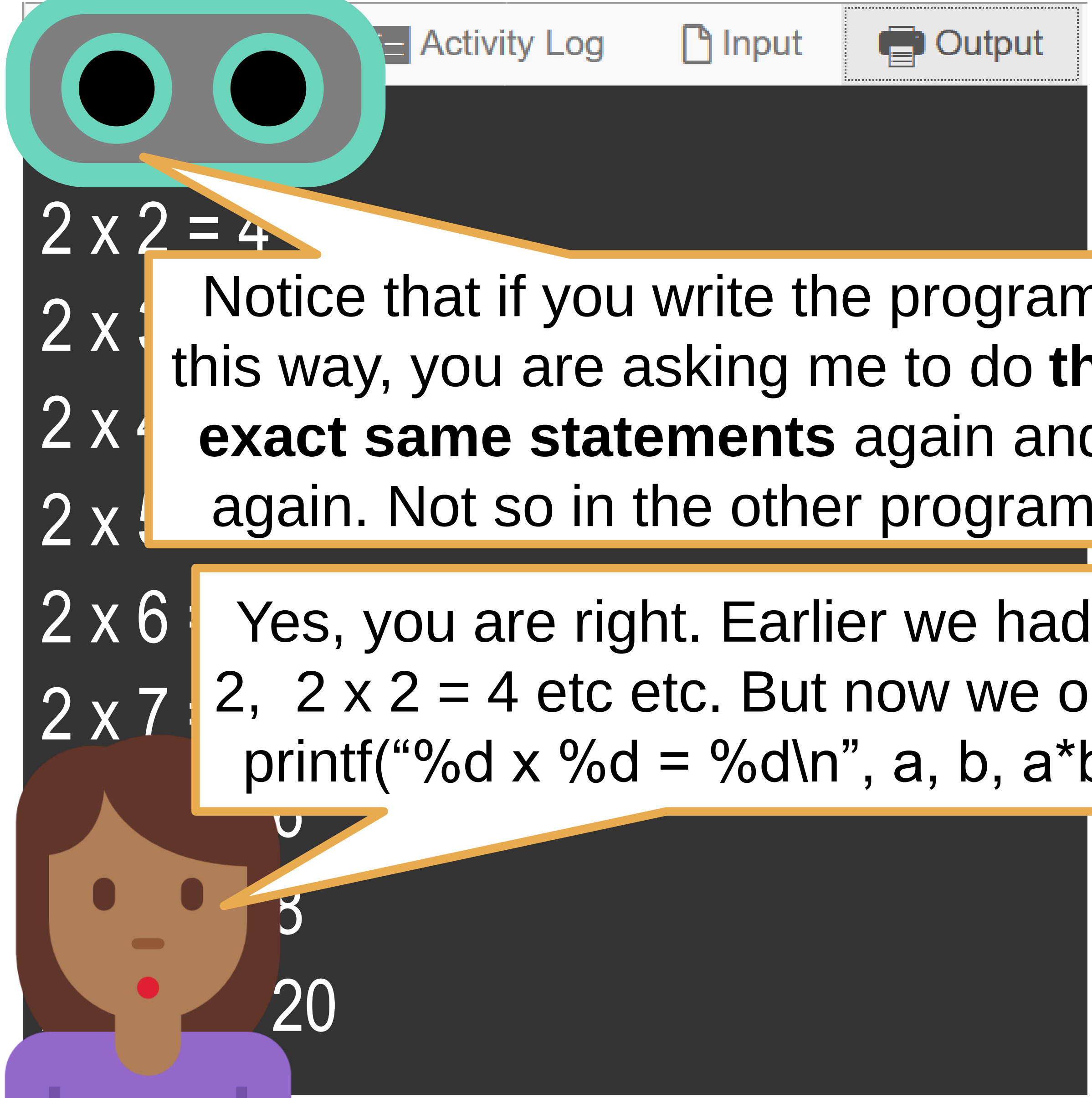
Notice that if you write the program this way, you are asking me to do **the exact same statements** again and again. Not so in the other program

```
int a = 2, b = 1;
printf(“%d x %d = %d\n”, a, b, a*b);
b++;
printf(“%d x %d = %d\n”, a, b, a*b);
b++;
printf(“%d x %d = %d\n”, a, b, a*b);
b++;
printf(“%d x %d = %d\n”, a, b, a*b
b++;
…
```

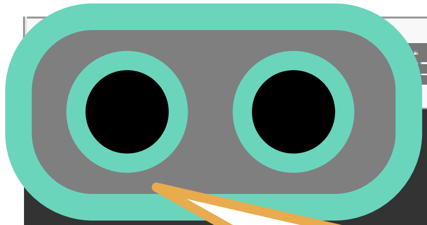
Yes, but I got bored *writing* them ☹



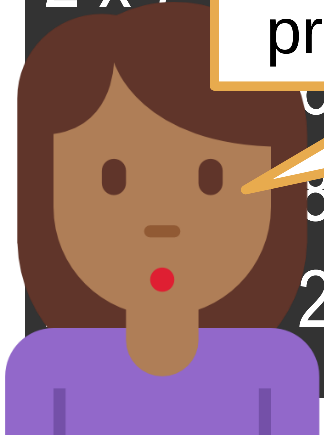
Printing the multiplication table of 2



```
2 x 2 = 4
2 x
2 x
2 x
2 x 6
2 x 7
6
8
20
```



Notice that if you write the program this way, you are asking me to do **the exact same statements** again and again. Not so in the other program



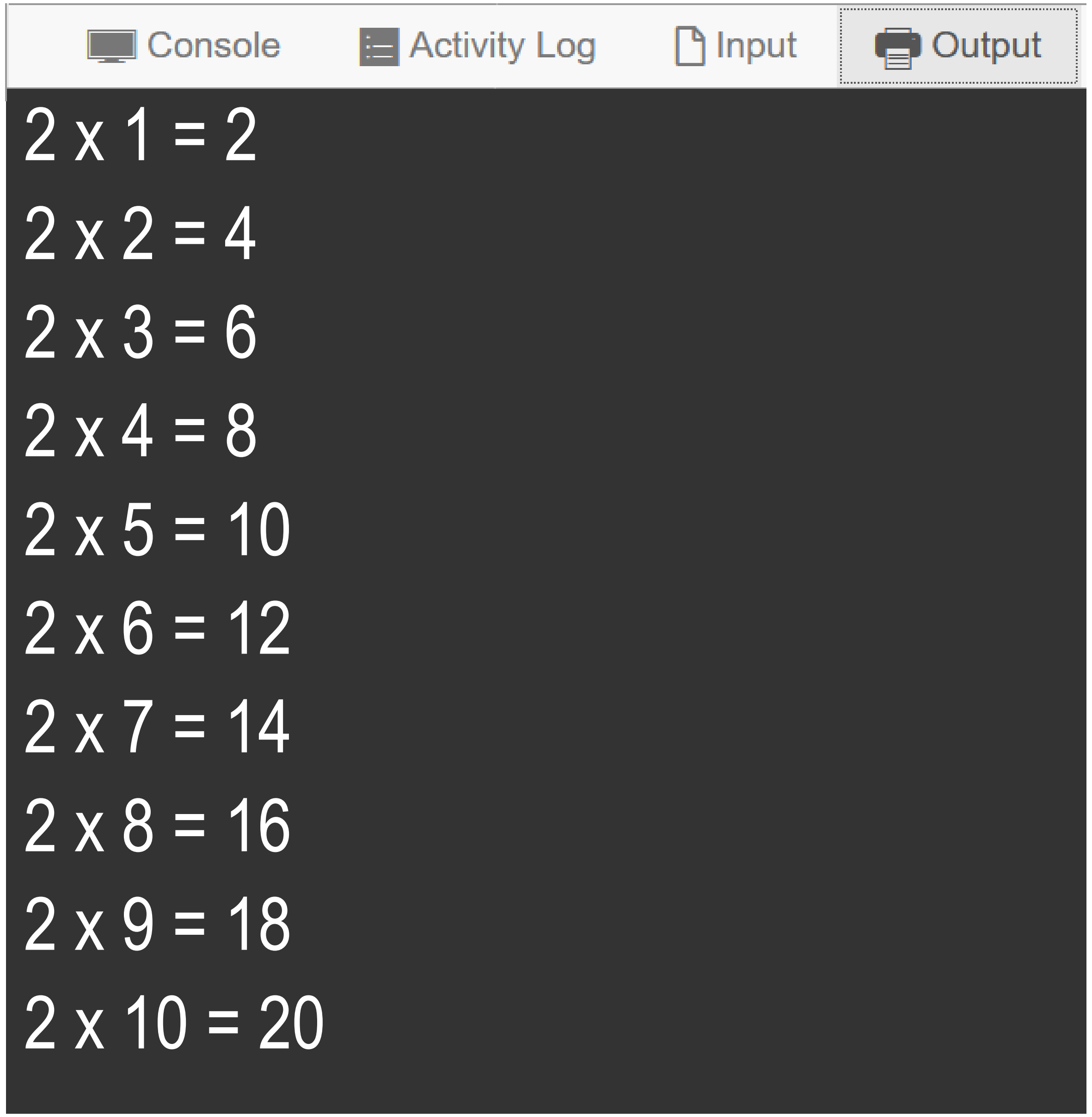
Yes, you are right. Earlier we had 2 x 1 = 2, 2 x 2 = 4 etc etc. But now we only have printf(“%d x %d = %d\n”, a, b, a*b); b++;

```
int a = 2, b = 1;
printf(“%d x %d = %d\n”, a, b, a*b);
b++;
printf(“%d x %d = %d\n”, a, b, a*b);
b++;
f(“%d x %d = %d\n”, a, b, a*b);

printf(“%d x %d = %d\n”, a, b, a*b
b++;
…
```

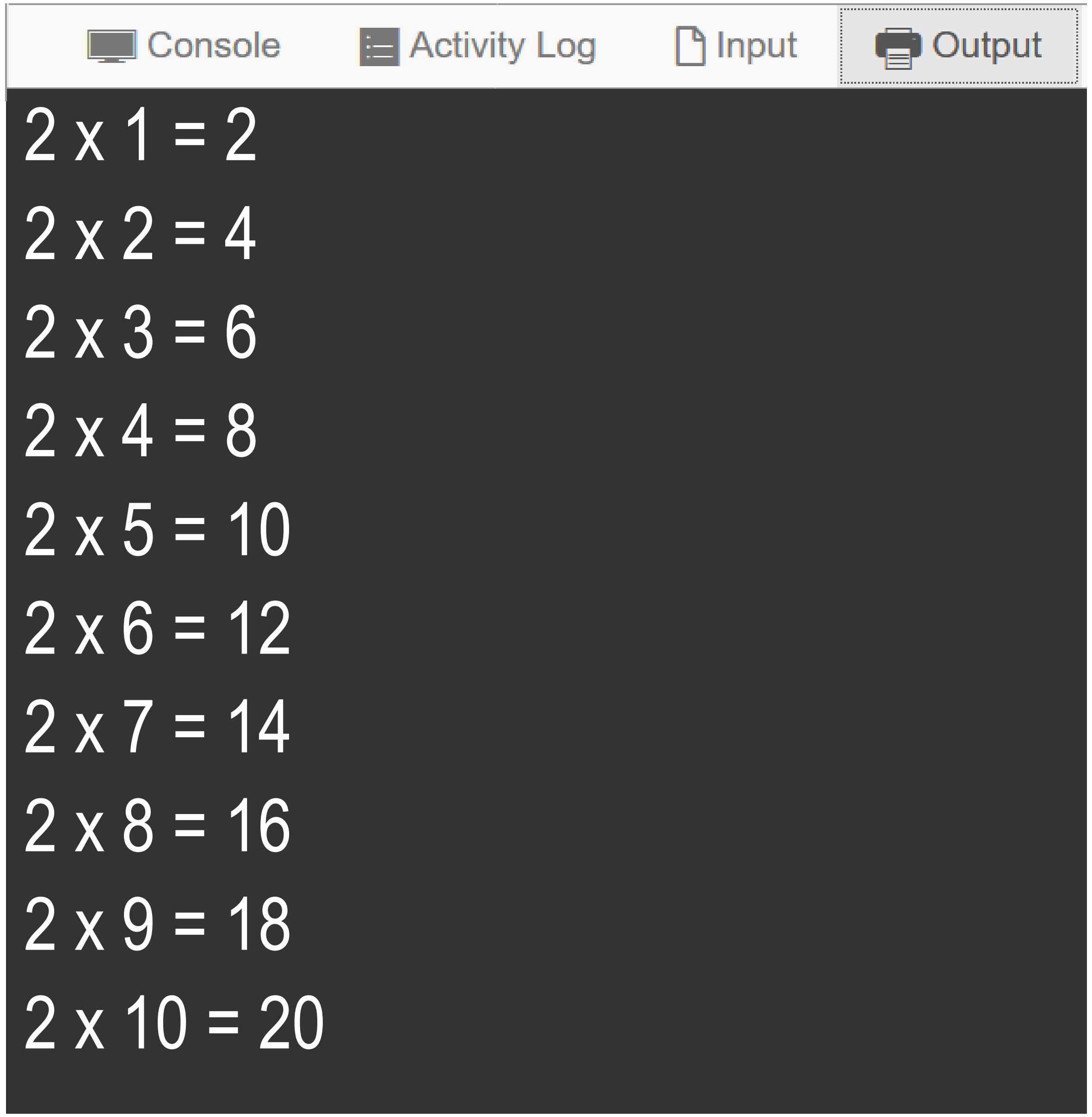
Yes, but I got bored *writing* them ☹

Printing the multiplication table of 2



```
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18
2 x 10 = 20
```

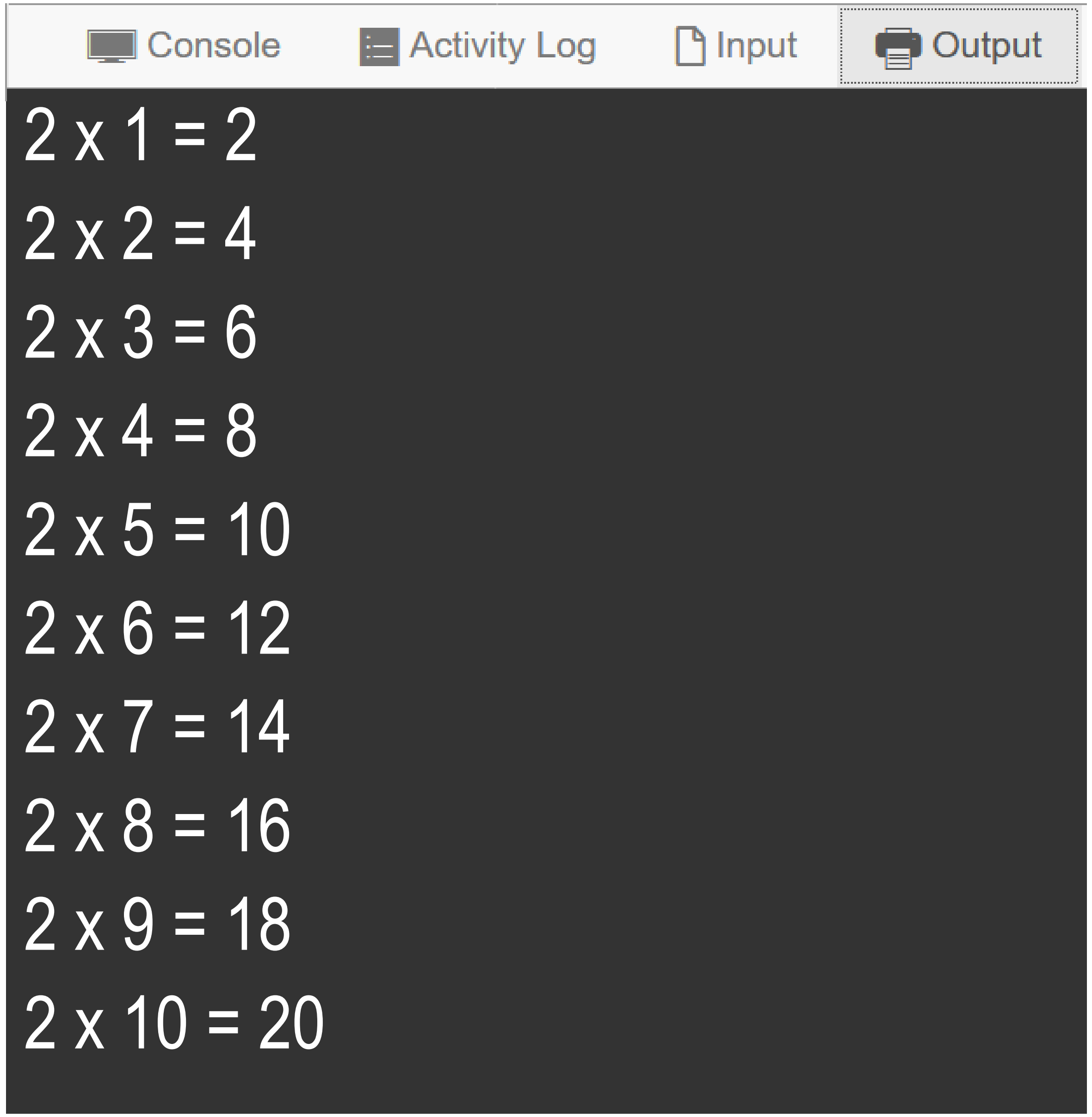
Printing the multiplication table of 2



```
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18
2 x 10 = 20
```

```c
int a = 2, b;
for(b = 1; b <= 10; b++){
    printf("%d x %d = %d\n", a, b, a*b);
}
```

Printing the multiplication table of 2

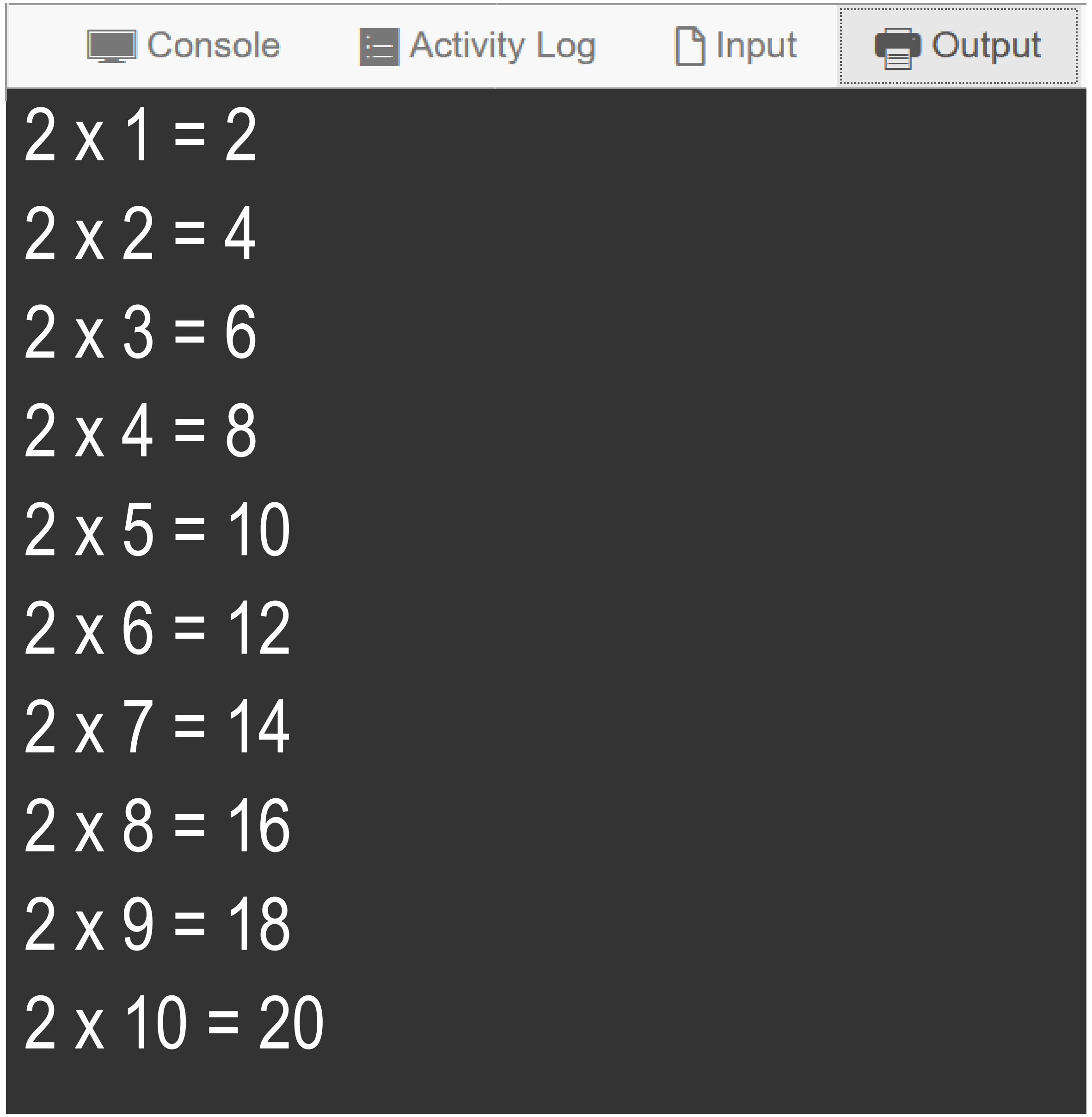


```
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18
2 x 10 = 20
```

```
int a = 2, b;
for(b = 1; b <= 10; b++){
    printf("%d x %d = %d\n", a, b, a*b);
}
```

How we usually speak to a human

Printing the multiplication table of 2

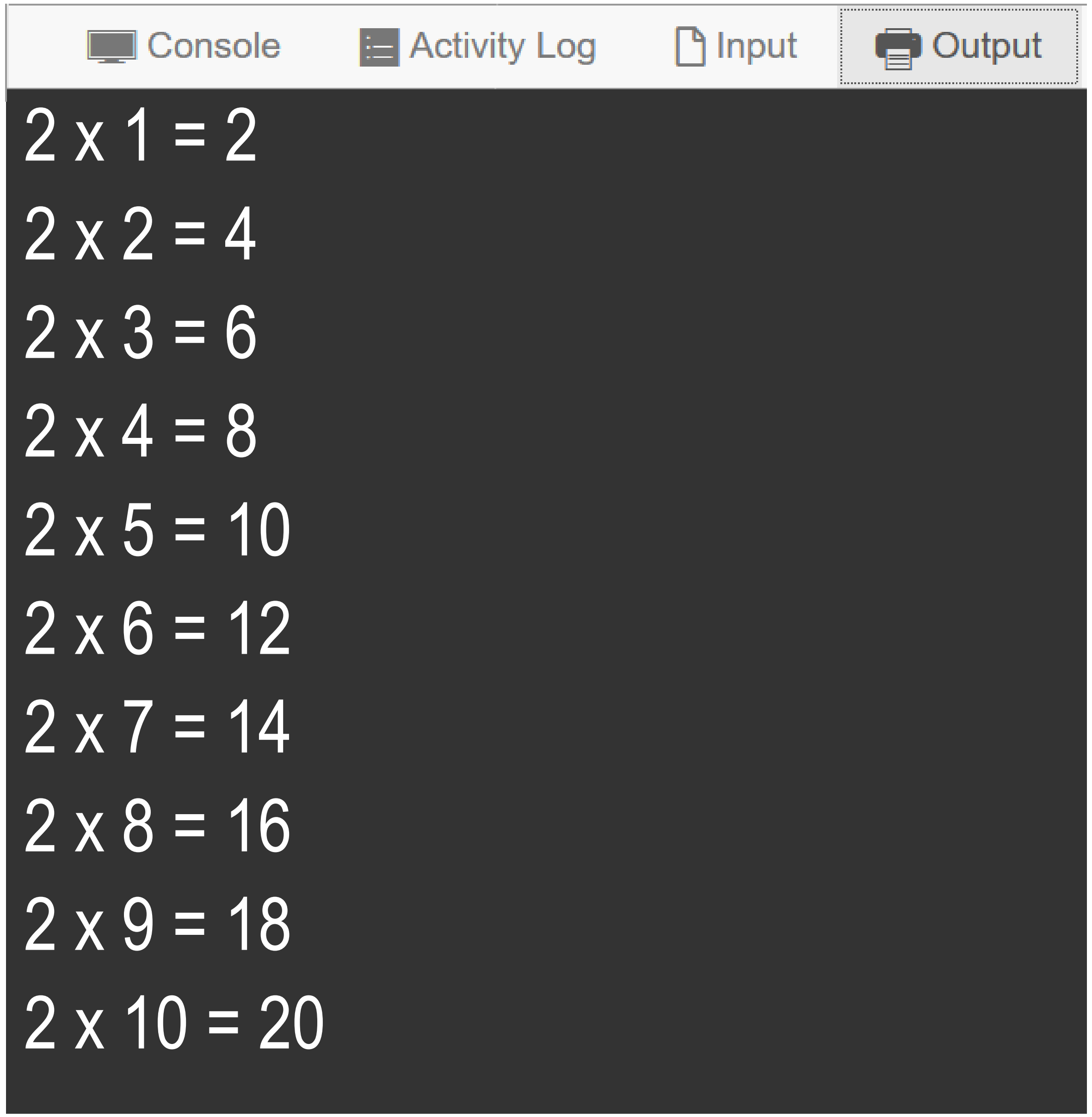


```
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18
2 x 10 = 20
```

```
int a = 2, b;
for(b = 1; b <= 10; b++){
    printf("%d x %d = %d\n", a, b, a*b);
}
```

How we usually speak to a human

Printing the multiplication table of 2



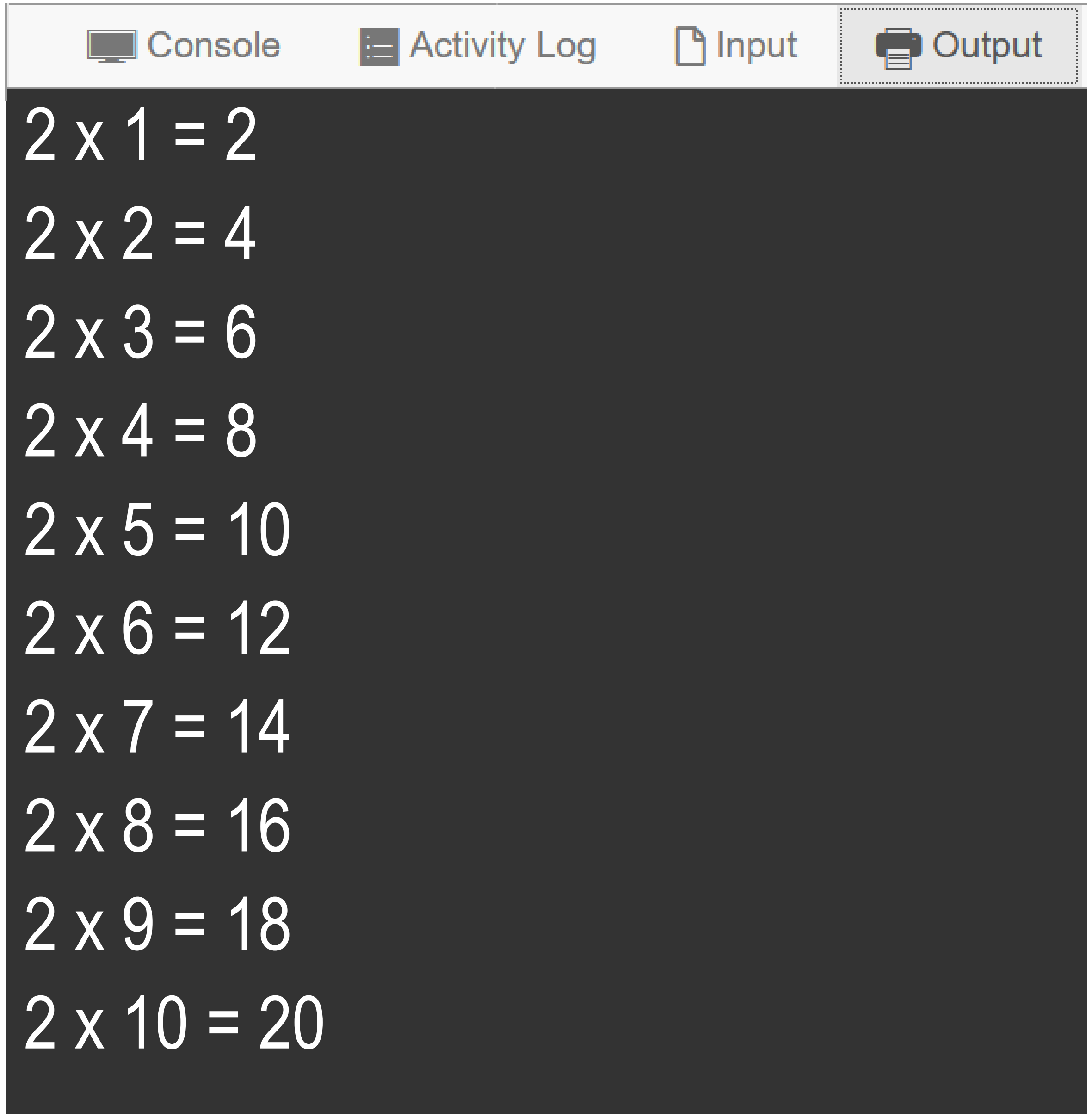
```
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18
2 x 10 = 20
```

```
int a = 2, b;
for(b = 1; b <= 10; b++){
    printf("%d x %d = %d\n", a, b, a*b);
}
```

How we usually speak to a human

1. Let a = 1, b be integer variables

Printing the multiplication table of 2



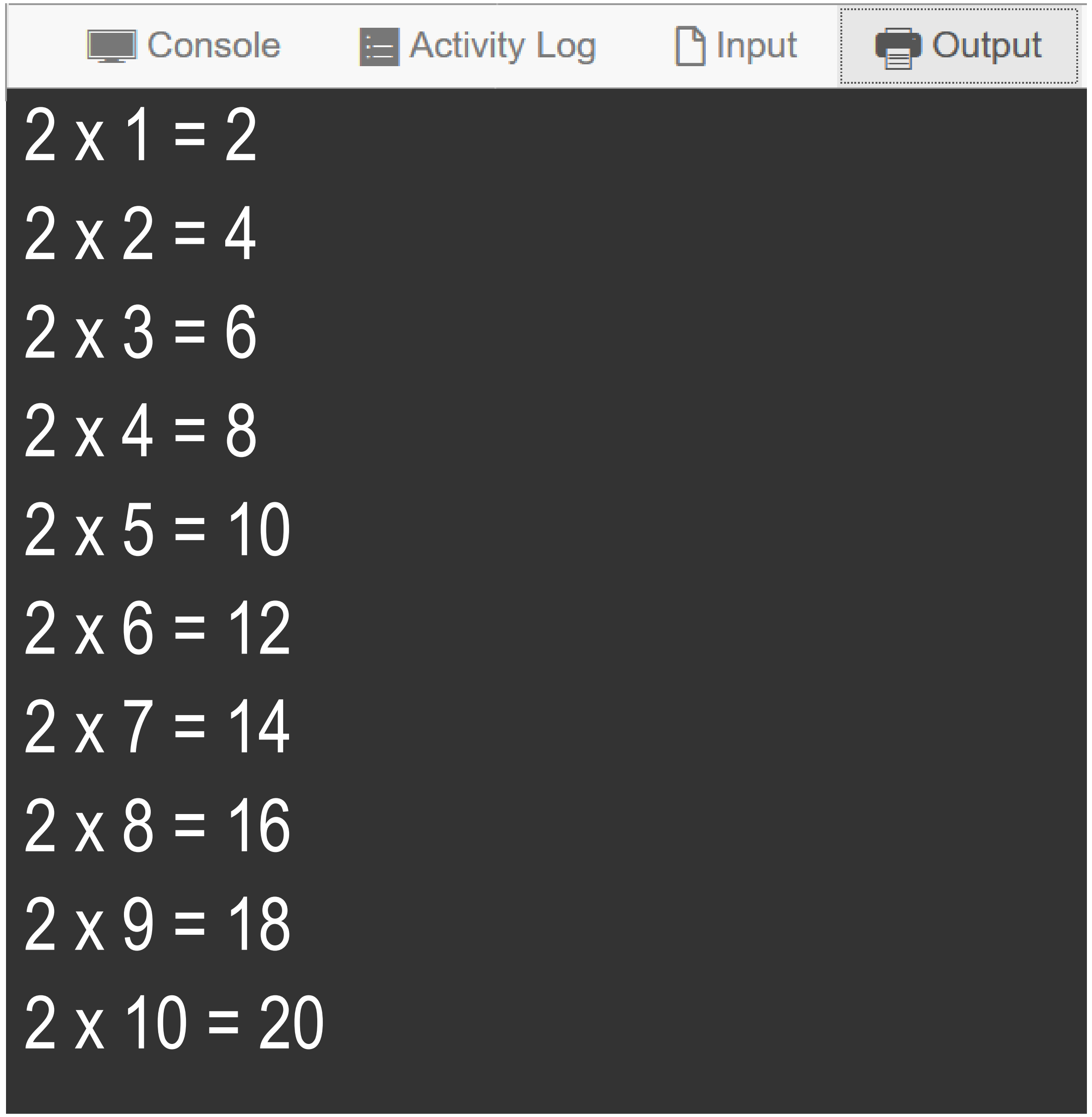
```
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18
2 x 10 = 20
```

```
int a = 2, b;
for(b = 1; b <= 10; b++){
    printf("%d x %d = %d\n", a, b, a*b);
}
```

How we usually speak to a human

1. Let a = 1, b be integer variables
2. First set b = 1

Printing the multiplication table of 2



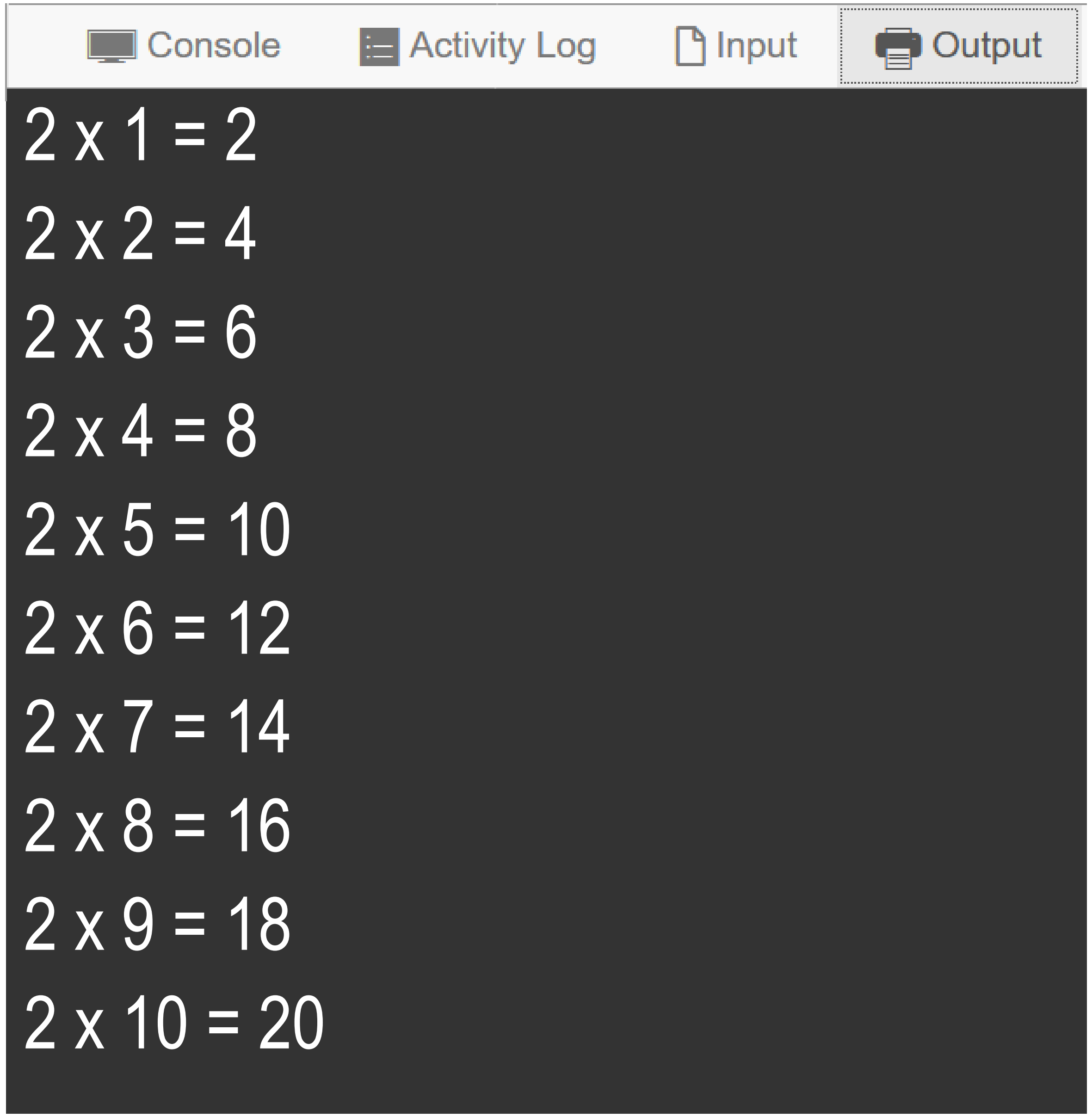
```
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18
2 x 10 = 20
```

```
int a = 2, b;
for(b = 1; b <= 10; b++){
    printf("%d x %d = %d\n", a, b, a*b);
}
```

How we usually speak to a human

1. Let a = 1, b be integer variables
2. First set b = 1
3. Then check if b <= 10 or not

Printing the multiplication table of 2



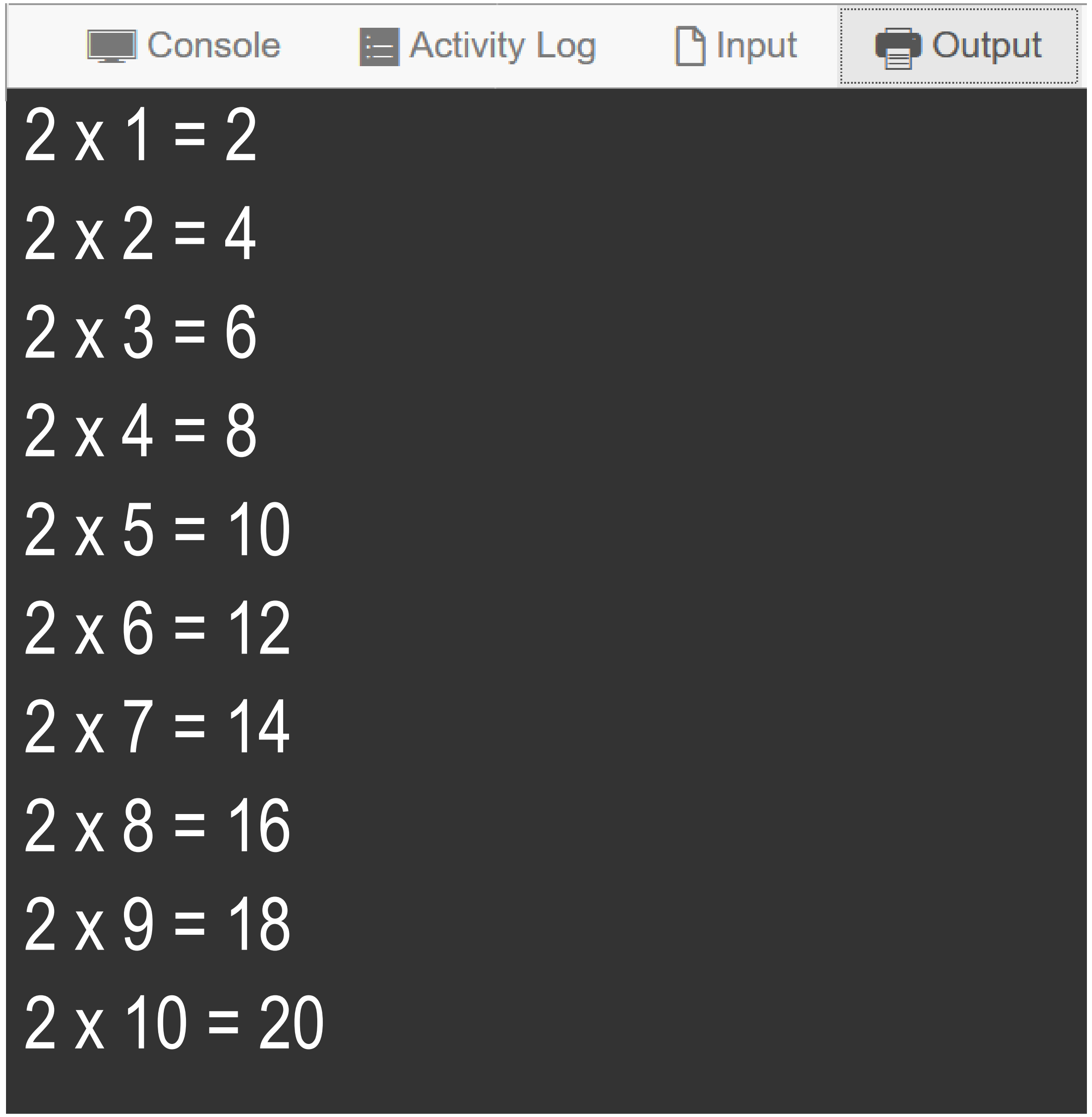
```
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18
2 x 10 = 20
```

```
int a = 2, b;
for(b = 1; b <= 10; b++){
    printf("%d x %d = %d\n", a, b, a*b);
}
```

How we usually speak to a human

1. Let a = 1, b be integer variables
2. First set b = 1
3. Then check if b <= 10 or not
    1. If true, execute printf statement, execute b++, go to step 3

Printing the multiplication table of 2



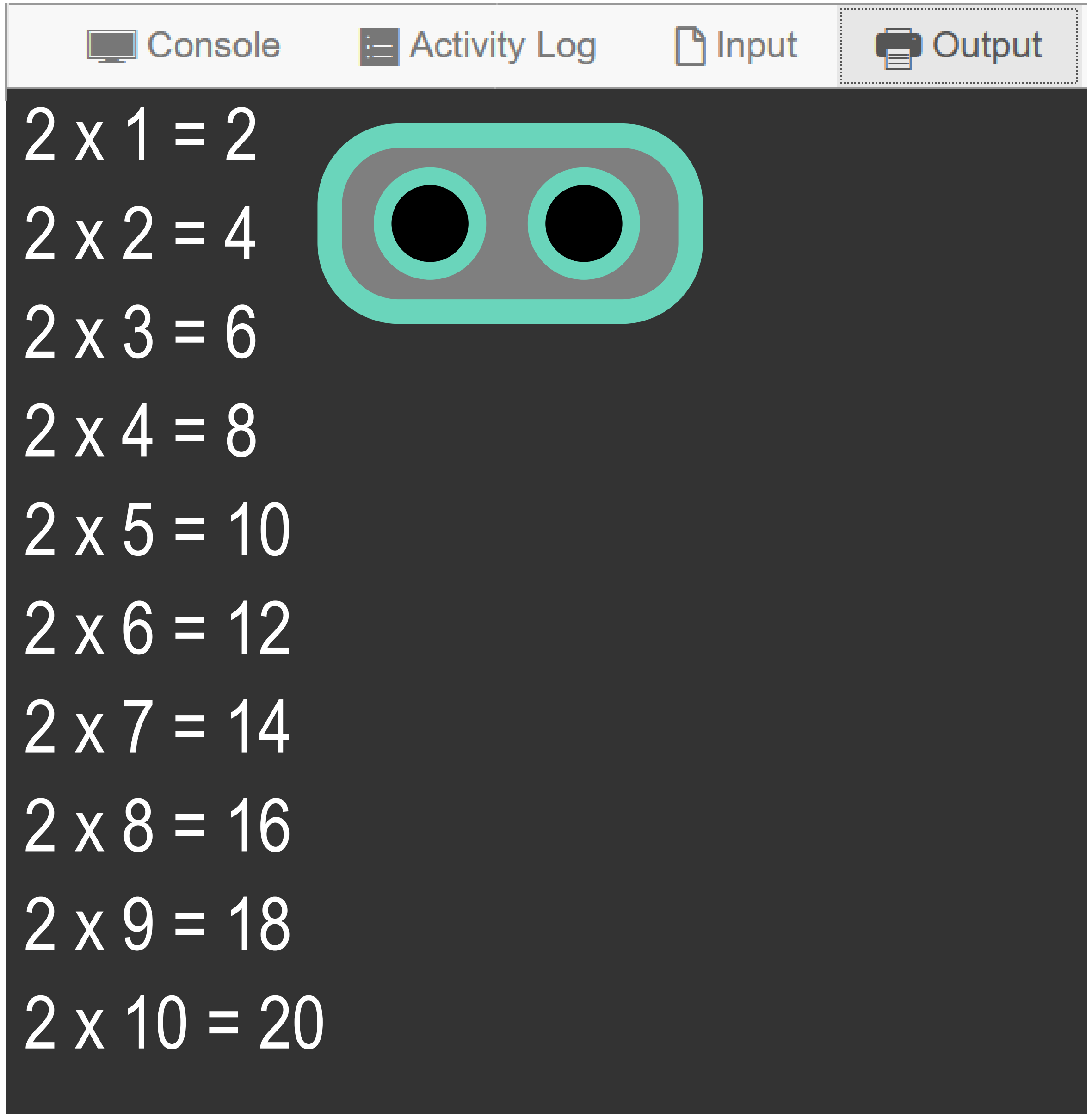
```
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18
2 x 10 = 20
```

```
int a = 2, b;
for(b = 1; b <= 10; b++){
    printf("%d x %d = %d\n", a, b, a*b);
}
```

How we usually speak to a human

1. Let a = 1, b be integer variables
2. First set b = 1
3. Then check if b <= 10 or not
   1. If true, execute printf statement, execute b++, go to step 3
   2. If false (i.e. b > 10), stop looping

Printing the multiplication table of 2



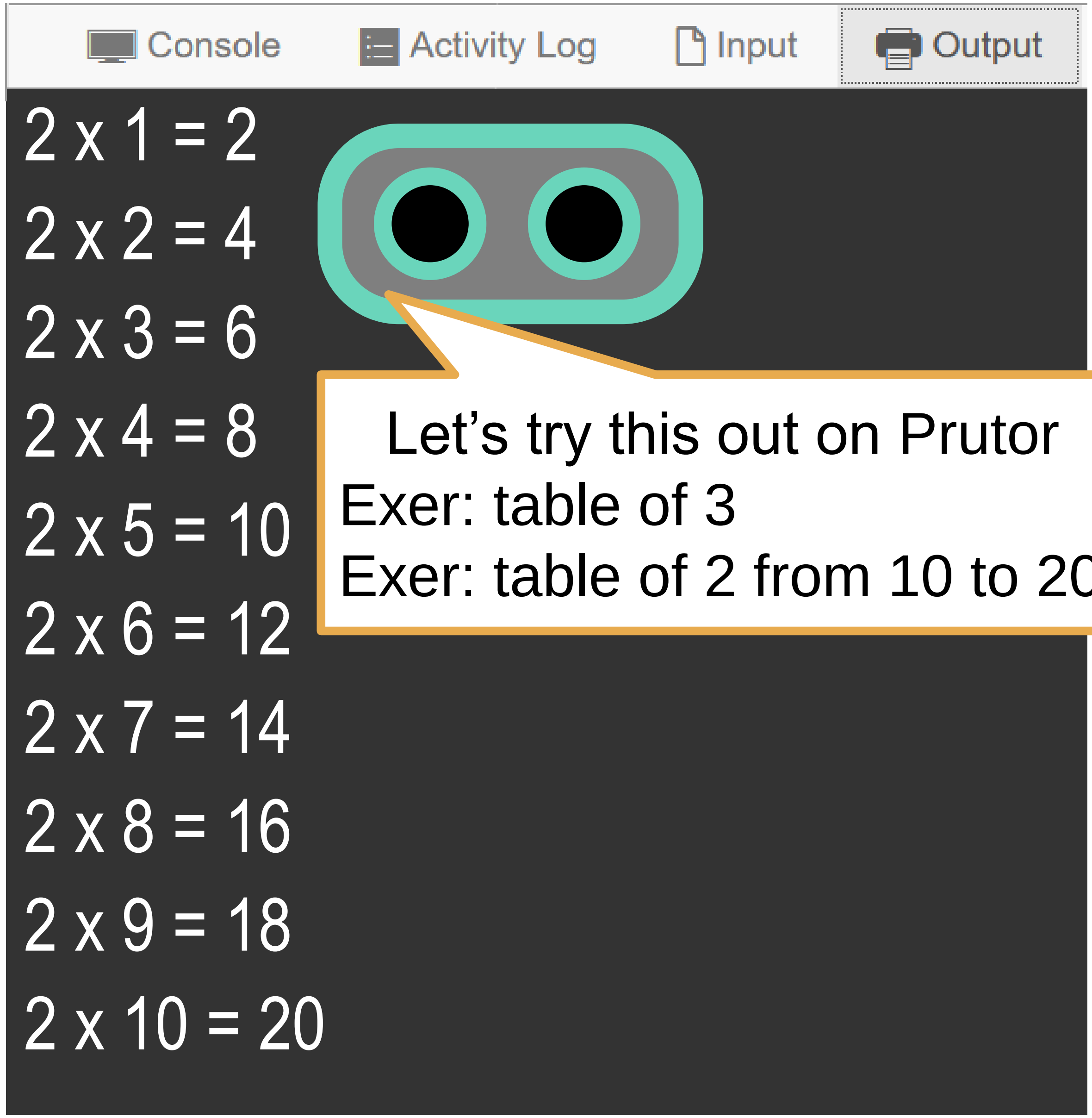
```
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18
2 x 10 = 20
```

```
int a = 2, b;
for(b = 1; b <= 10; b++){
    printf("%d x %d = %d\n", a, b, a*b);
}
```

How we usually speak to a human

1. Let a = 1, b be integer variables
2. First set b = 1
3. Then check if b <= 10 or not
   1. If true, execute printf statement, execute b++, go to step 3
   2. If false (i.e. b > 10), stop looping

Printing the multiplication table of 2



```
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18
2 x 10 = 20
```

Let's try this out on Prutor
Exer: table of 3
Exer: table of 2 from 10 to 20

```
int a = 2, b;
for(b = 1; b <= 10; b++){
    printf("%d x %d = %d\n", a, b, a*b);
}
```

How we usually speak to a human

1. Let a = 1, b be integer variables
2. First set b = 1
3. Then check if b <= 10 or not
   1. If true, execute printf statement, execute b++, go to step 3
   2. If false (i.e. b > 10), stop looping

The for loop

The for loop

General form of a for loop

The for loop

General form of a for loop

```
for(init_expr; stopping_expr; update_expr){
    statement1;
    statement2;
    ...
}
statement3;
statement4;
...
```

The for loop

General form of a for loop

```
for(init_expr; stopping_expr; update_expr){
    statement1;
    statement2;
    ...
}
statement3;
statement4;
...
```

How we usually speak to a human

The for loop

General form of a for loop

```
for(init_expr; stopping_expr; update_expr){
    statement1;
    statement2;
    ...
}
statement3;
statement4;
...
```

How we usually speak to a human

The for loop

General form of a for loop

```
for(init_expr; stopping_expr; update_expr){
    statement1;
    statement2;
    ...
}
statement3;
statement4;
...
```

How we usually speak to a human

1. First do what is told in initialization expression

The for loop

General form of a for loop

```
for(init_expr; stopping_expr; update_expr){
    statement1;
    statement2;
    ...
}
statement3;
statement4;
...
```

How we usually speak to a human

1. First do what is told in initialization expression

The for loop

General form of a for loop

```
for(init_expr; stopping_expr; update_expr){
    statement1;
    statement2;
    …
}
statement3;
statement4;
…
```

How we usually speak to a human

1. First do what is told in initialization expression
2. Then check the stopping expression

The for loop

General form of a for loop

```
for(init_expr; stopping_expr; update_expr){
    statement1;
    statement2;
    ...
}
statement3;
statement4;
...
```

How we usually speak to a human

1. First do what is told in initialization expression
2. Then check the stopping expression

The for loop

General form of a for loop

```
for(init_expr; stopping_expr; update_expr){
    statement1;
    statement2;
    ...
}
statement3;
statement4;
...
```

How we usually speak to a human

1. First do what is told in initialization expression
2. Then check the stopping expression
3. If stopping expression is true

The for loop

General form of a for loop

```
for(init_expr; stopping_expr; update_expr){
    statement1;
    statement2;
    ...
}
statement3;
statement4;
...
```

How we usually speak to a human

1. First do what is told in initialization expression
2. Then check the stopping expression
3. If stopping expression is true
   Execute all statements inside braces

The for loop

General form of a for loop

```
for(init_expr; stopping_expr; update_expr){
    statement1;
    statement2;
    ...
}
statement3;
statement4;
...
```

How we usually speak to a human

1. First do what is told in initialization expression
2. Then check the stopping expression
3. If stopping expression is true
   Execute all statements inside braces

The for loop

General form of a for loop

```
for(init_expr; stopping_expr; update_expr){
    statement1;
    statement2;
    ...
}
statement3;
statement4;
...
```

How we usually speak to a human

1. First do what is told in initialization expression
2. Then check the stopping expression
3. If stopping expression is true
   Execute all statements inside braces
   Execute update expression

The for loop

General form of a for loop

```
for(init_expr; stopping_expr; update_expr){
    statement1;
    statement2;
    ...
}
statement3;
statement4;
...
```

How we usually speak to a human

1. First do what is told in initialization expression
2. Then check the stopping expression
3. If stopping expression is true
    Execute all statements inside braces
    Execute update expression

The for loop

General form of a for loop

```
for(init_expr; stopping_expr; update_expr){
    statement1;
    statement2;
    ...
}
statement3;
statement4;
...
```

How we usually speak to a human

1. First do what is told in initialization expression
2. Then check the stopping expression
3. If stopping expression is true
   - Execute all statements inside braces
   - Execute update expression
   - Go back to step 2

The for loop

General form of a for loop

```
for(init_expr; stopping_expr; update_expr){
    statement1;
    statement2;
    ...
}
statement3;
statement4;
...
```

How we usually speak to a human

1. First do what is told in initialization expression
2. Then check the stopping expression
3. If stopping expression is true
    Execute all statements inside braces
    Execute update expression
    Go back to step 2
   Else stop looping and execute rest of code

The for loop

General form of a for loop

```
for(init_expr; stopping_expr; update_expr){
    statement1;
    statement2;
    ...
}
statement3;
statement4;
...
```

How we usually speak to a human

1. First do what is told in initialization expression
2. Then check the stopping expression
3. If stopping expression is true
    - Execute all statements inside braces
    - Execute update expression
    - Go back to step 2

   Else stop looping and execute rest of code

The for loop

General form of a for loop

```
for(init_expr; stopping_expr; update_expr){
    statement1;
    statement2;
    …
}
statement3;
statement4;
…
```

How we usually speak to a human

1. First do what is told in initialization expression
2. Then check the stopping expression
3. If stopping expression is true
    - Execute all statements inside braces
    - Execute update expression
    - Go back to step 2

   Else stop looping and execute rest of code

The for loop

General form of a for loop

```
for(init_expr; stopping_expr; update_expr){
    statement1;
    statement2;
    ...
}
statement3;
statement4;
...
```

How we usually speak to a human

1. First do what is told in initialization expression
2. Then check the stopping expression
3. If stopping expression is true
   Execute all statements inside braces
   Execute update expression
   Go back to step 2
   Else stop looping and execute rest of code

The for loop

Brackets essential if you want me to do many things while looping

General form of a for loop

```
for(init_expr; stopping_expr; update_expr){
    statement1;
    statement2;
    ...
}
statement3;
statement4;
...
```

How we usually speak to a human

1. First do what is told in initialization expression
2. Then check the stopping expression
3. If stopping expression is true
    - Execute all statements inside braces
    - Execute update expression
    - Go back to step 2

   Else stop looping and execute rest of code

The for loop

The for loop

```
for(init_expr; stopping_expr; update_expr){
    statement1;
    statement2;
}
```

The for loop

```
for(init_expr; stopping_expr; update_expr){
    statement1;
    statement2;
 }
```

The entire for loop is considered one statement

The for loop

```
for(init_expr; stopping_expr; update_expr){
    statement1;
    statement2;
 }
```

The entire for loop is considered one statement

The for loop

```
for(init_expr; stopping_expr; update_expr){
    statement1;
    statement2;
}
```

The entire for loop is considered one statement

Can put inside for loops: printf statements, if-else/switch statements, even for loop statement (nested for loop)

The for loop

```
for(init_expr; stopping_expr; update_expr){
    statement1;
    statement2;
}
```

The entire for loop is considered one statement

Can put inside for loops: printf statements, if-else/switch statements, even for loop statement (nested for loop)

Usually init_expr, stopping_expr, update_expr involve the same variable, e.g. b in multiplication table example

The for loop

```
for(init_expr; stopping_expr; update_expr){
    statement1;
    statement2;
}
```

The entire for loop is considered one statement

Can put inside for loops: printf statements, if-else/switch statements, even for loop statement (nested for loop)

Usually init_expr, stopping_expr, update_expr involve the same variable, e.g. b in multiplication table example

Lovingly called variable of the loop/counter variable

The for loop

```
for(init_expr; stopping_expr; update_expr){
    statement1;
    statement2;
}
```

The for loop

```
for(init_expr; stopping_expr; update_expr){
    statement1;
    statement2;
 }
```

stopping_expr must give true/false value

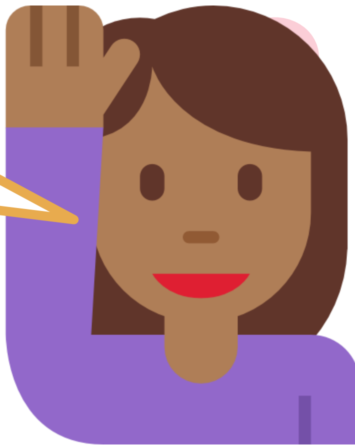
The for loop

```
for(init_expr; stopping_expr; update_expr){
    statement1;
    statement2;
}
```

stopping_expr must give true/false value

The for loop

All expressions generate values, even assignment/relational ones

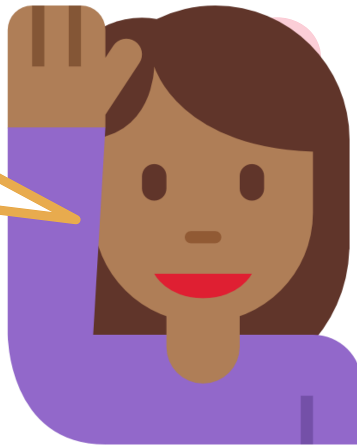


```
for(init_expr; stopping_expr; update_expr){
    statement1;
    statement2;
}
```

stopping_expr must give true/false value

The for loop

All expressions generate values, even assignment/relational ones



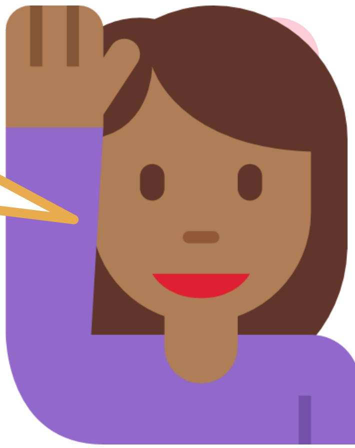
```
for(init_expr; stopping_expr; update_expr){
    statement1;
    statement2;
}
```

stopping_expr must give true/false value

Usually done by making stopping_expr a relational expression

The for loop

All expressions generate values, even assignment/relational ones



```
for(init_expr; stopping_expr; update_expr){
    statement1;
    statement2;
 }
```

stopping_expr must give true/false value

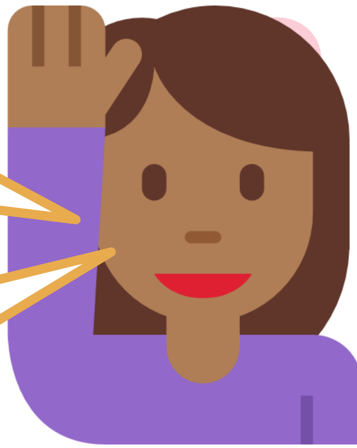
- Usually done by making stopping_expr a relational expression
- Warning: you can say b * 2 in stopping_expr but dangerous

The for loop

All expressions generate values, even assignment/relational ones

```
for(init_expr; stopping_exp
    statement1;
    statement2;
}
```

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



stopping_expr must give true/false value

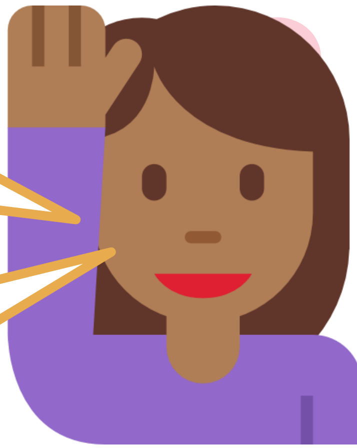
- Usually done by making stopping_expr a relational expression
- Warning: you can say b * 2 in stopping_expr but dangerous

The for loop

```
for(init_expr; stopping_exp
    statement1;
    statement2;
}
```

All expressions generate values, even assignment/relational ones

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



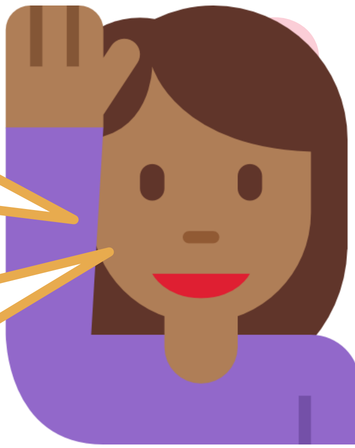
- stopping_expr must give true/false value
  - Usually done by making stopping_expr a relational expression
  - Warning: you can say b * 2 in stopping_expr but dangerous
  - init_expr and update_expr can be anything you want

The for loop

```
for(init_expr; stopping_exp
    statement1;
    statement2;
}
```

All expressions generate values, even assignment/relational ones

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

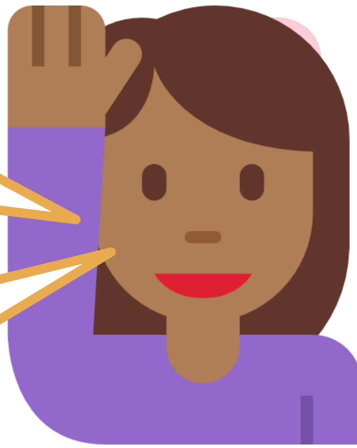


- stopping_expr must give true/false value
  - Usually done by making stopping_expr a relational expression
  - Warning: you can say b * 2 in stopping_expr but dangerous
  - init_expr and update_expr can be anything you want
- init_expr and update_expr can even be empty

The for loop

All expressions generate values, even assignment/relational ones

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



```
for(init_expr; stopping_exp
    statement1;
    statement2;
}
```

stopping_expr must give true/false value

- Usually done by making stopping_expr a relational expression
- Warning: you can say b * 2 in stopping_expr but dangerous
- init_expr and update_expr can be anything you want

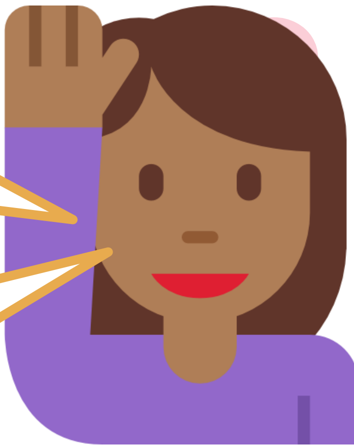
init_expr and update_expr can even be empty

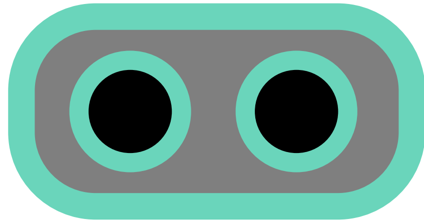
```
for(;stopping_expr;){ ... }
```

The for loop

All expressions generate values, even assignment/relational ones

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





```
for(init_expr; stopping_exp
    statement1;
    statement2;
}
```

stopping_expr must give true/false value

- Usually done by making stopping_expr a relational expression
- Warning: you can say b * 2 in stopping_expr but dangerous
- init_expr and update_expr can be anything you want

init_expr and update_expr can even be empty

```
for(;stopping_expr;){ ... }
```

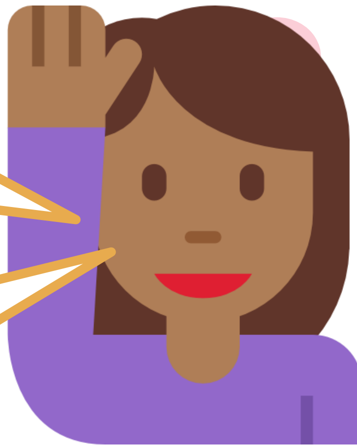
The for loop

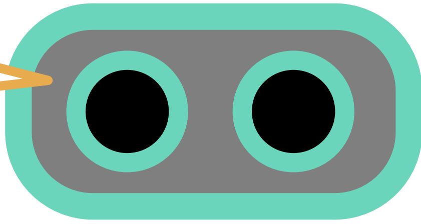
```
for(init_expr; stopping_exp
    statement1;
    statement2;
}
```

All expressions generate values, even assignment/relational ones

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

Yes, you can write the init_expr before the loop and the update_expr inside the loop





stopping_expr must give true/false value

- Usually done by making stopping_expr a relational expression
- Warning: you can say b * 2 in stopping_expr but dangerous
- init_expr and update_expr can be anything you want

init_expr and update_expr can even be empty

```
for(;stopping_expr;){ ... }
```

The for loop

```
for(init_expr; stopping_exp
    statement1;
    statement2;
}
```

stopping_expr must give true/false value

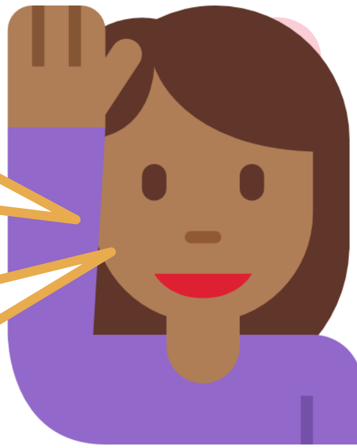
Usually done by making stopping_expr a relational ex

Warning: you can say b * 2 in stopping_expr but dangerous

init_expr and update_expr can be anything you want

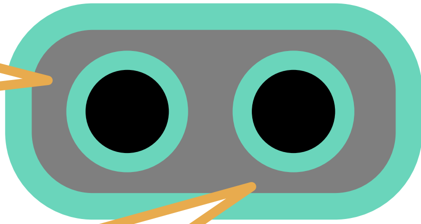
init_expr and update_expr can even be empty

```
for(;stopping_expr;){ ... }
```



All expressions generate values, even assignment/relational ones

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



Yes, you can write the init_expr before the loop and the update_expr inside the loop

By the way, even the stopping_expr can be empty

The for loop

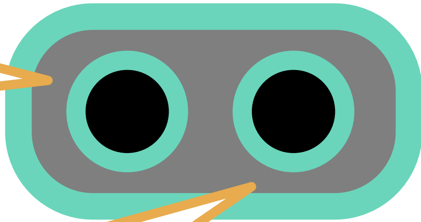
```
for(init_expr; stopping_exp
    statement1;
    statement2;
}
```

All expressions generate values, even assignment/relational ones

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

Yes, you can write the init_expr before the loop and the update_expr inside the loop

By the way, even the stopping_expr can be empty



- stopping_expr must give true/false value
  - Usually done by making stopping_expr a relational ex
  - Warning: you can say b * 2 in stopping_expr but dangerous
  - init_expr and update_expr can be anything you want
- init_expr and update_expr can even be empty

```
for(;stopping_expr;){ ... }
```

The for loop

```
for(init_expr; stopping_exp
    statement1;
    statement2;
}
```

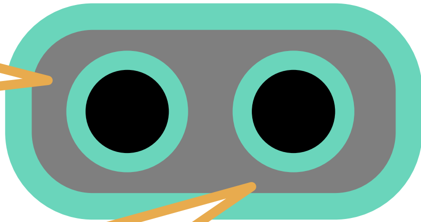
All expressions generate values, even assignment/relational ones

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

Yes, you can write the init_expr before the loop and the update_expr inside the loop

By the way, even the stopping_expr can be empty

Next class ☺



- stopping_expr must give true/false value
  - Usually done by making stopping_expr a relational ex
  - Warning: you can say b * 2 in stopping_expr but dangero
  - init_expr and update_expr can be anything you want
- init_expr and update_expr can even be empty

```
for(;stopping_expr;){ ... }
```

Some common errors in loops

Some common errors in loops

Initialization: forget to do it or else wrong initialization

Some common errors in loops

Initialization: forget to do it or else wrong initialization

Statements: Note, update_expr executed **after** statements

Some common errors in loops

Initialization: forget to do it or else wrong initialization

Statements: Note, update_expr executed **after** statements

Update: Forget to do update step or wrong update step

Some common errors in loops

Initialization: forget to do it or else wrong initialization

Statements: Note, update_expr executed **after** statements

Update: Forget to do update step or wrong update step

Termination: wrong or missing termination

Some common errors in loops

Initialization: forget to do it or else wrong initialization

Statements: Note, update_expr executed **after** statements

Update: Forget to do update step or wrong update step

Termination: wrong or missing termination

for(b=1;**b<10**;b++){…} not same as for(b=1;**b<=10**;b++){…}

Some common errors in loops

Initialization: forget to do it or else wrong initialization

Statements: Note, update_expr executed **after** statements

Update: Forget to do update step or wrong update step

Termination: wrong or missing termination

for(b=1;**b<10**;b++){…} not same as for(b=1;**b<=10**;b++){…}

Infinite loop: The loop goes on forever. Never terminates.

Some common errors in loops

Initialization: forget to do it or else wrong initialization

Statements: Note, update_expr executed **after** statements

Update: Forget to do update step or wrong update step

Termination: wrong or missing termination

for(b=1;**b<10**;b++){…} not same as for(b=1;**b<=10**;b++){…}

Infinite loop: The loop goes on forever. Never terminates.

for(b=2;b>=1,b++){…}

Some common errors in loops

Initialization: forget to do it or else wrong initialization

Statements: Note, update_expr executed **after** statements

Update: Forget to do update step or wrong update step

Termination: wrong or missing termination

for(b=1;**b<10**;b++){…} not same as for(b=1;**b<=10**;b++){…}

Infinite loop: The loop goes on forever. Never terminates.

for(b=2;b>=1,b++){…}

Prutor will give "TLE" error (time limit exceeded error)