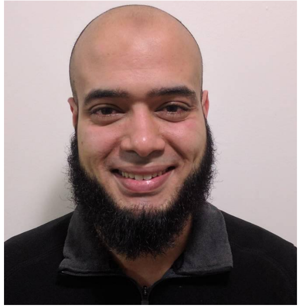
Programming4kids

While Loops

Mostafa Saad Ibrahim
Computer Vision Researcher @ Huawei Canada
PhD - Simon Fraser University
Bachelor / Msc - FCI Cairo University

Ex-(Software Engineer / Teaching Assistant)

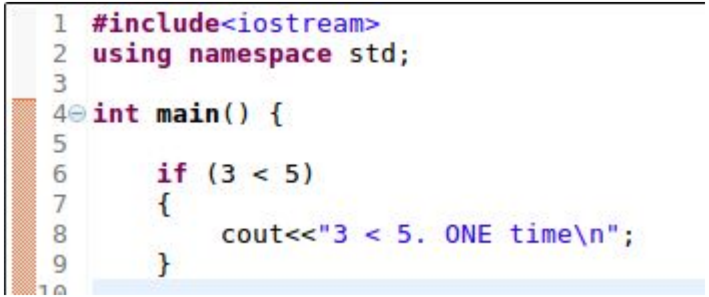


Loop

Src

- Loop means circular.
- In our context, it is a way to command the computer to
  - **REPEAT** the same operations till we ask to **STOP**

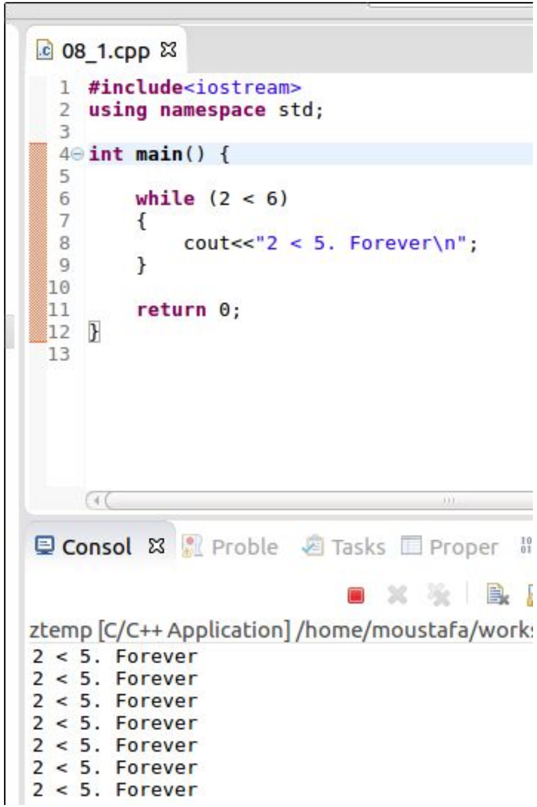
Recall if condition



```
#include<iostream>
using namespace std;

int main() {

    if (3 < 5)
    {
        cout<<"3 < 5. ONE time\n";
    }
```

- This if statement is applied once
- What if we want it repeat for ever?

Repeat for EVER



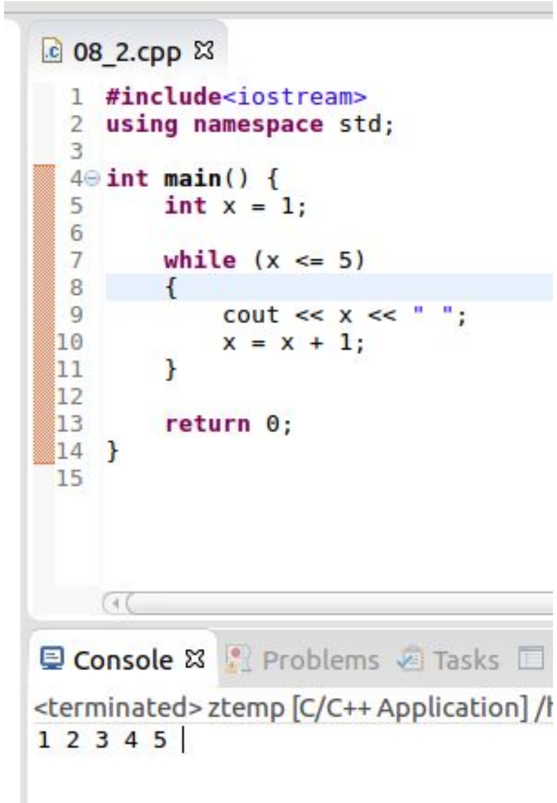
```
 1 #include<iostream>
 2 using namespace std;
 3
 4 int main() {
 5
 6     while (2 < 6)
 7     {
 8         cout<<"2 < 5. Forever\n";
 9     }
10
11     return 0;
12 }
13
```

ztemp [C/C++ Application] /home/moustafa/work:

```
2 < 5. Forever
2 < 5. Forever
2 < 5. Forever
2 < 5. Forever
2 < 5. Forever
2 < 5. Forever
2 < 5. Forever
```

- While is same logic, but code goes as following
  - Line 6
  - Line 8
  - Line 6
  - Line 8
  - Line 6
  - Line 8
  - For ever
- Notice the red button in eclipse
  - It means the program is still running!
  - We have to click it to force stop!
- What if wanna it STOP at some time?
  - Change while (condition)

Let’s print first 5 numbers

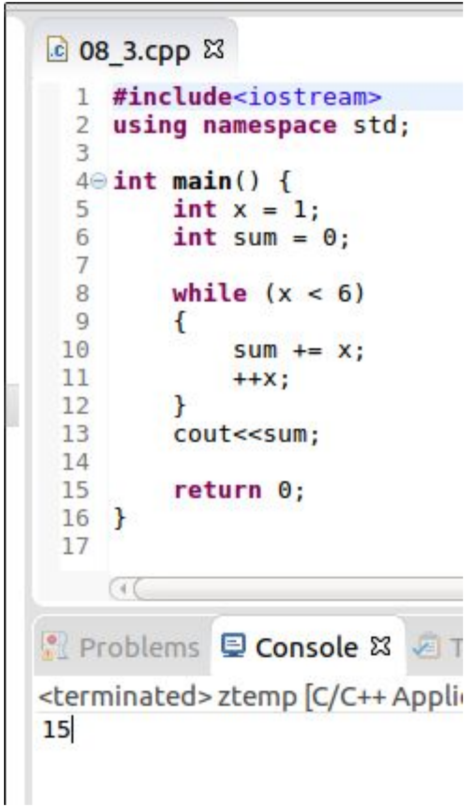


```
 1 #include<iostream>
 2 using namespace std;
 3 
 4 int main() {
 5     int x = 1;
 6 
 7     while (x <= 5)
 8     {
 9         cout << x << " ";
10         x = x + 1;
11     }
12 
13     return 0;
14 }
15 
```

```
<terminated> ztemp [C/C++ Application] /
1 2 3 4 5
```

- Body from lines 7 to 11 repeats as LONG as condition is true (x <= 5)
- Let’s trace it
- 3 important parts
    - Line 5 = Initialization
    - Line 7 = Condition
    - Line 10 = increment step

Let’s sum 1 + 2 + 3 + 4 + 5

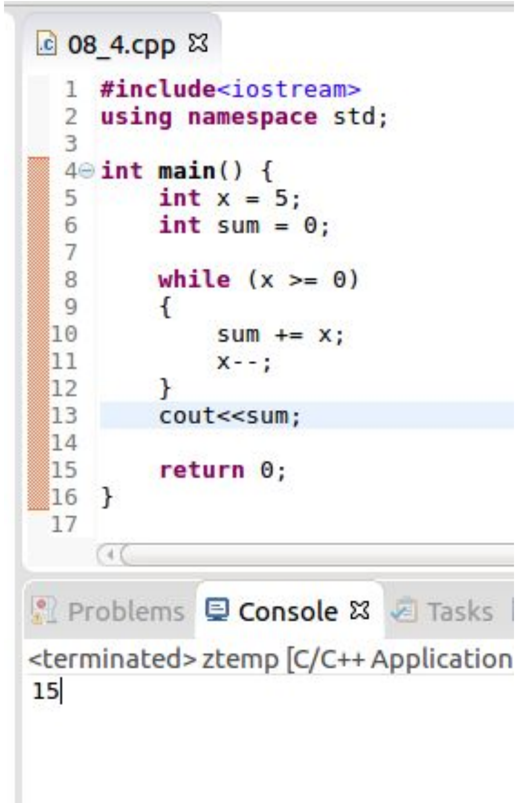


```
#include<iostream>
using namespace std;

int main() {
    int x = 1;
    int sum = 0;

    while (x < 6)
    {
        sum += x;
        ++x;
    }
    cout<<sum;

    return 0;
}

```

```
<terminated> ztemp [C/C++ Appli
15
```

- We can now make any complex logic that requires repetition!
- Let’s trace the code

Let’s sum 5 + 4 + 3 + 2 + 1

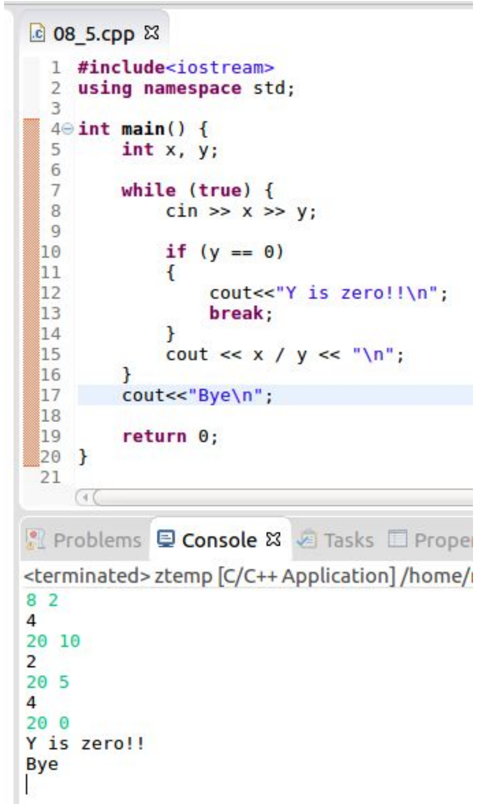


```
#include<iostream>
using namespace std;

int main() {
	int x = 5;
	int sum = 0;

	while (x >= 0)
	{
		sum += x;
		x--;
	}
	cout<<sum;

	return 0;
}
```

```
<terminated> ztemp [C/C++ Application
15
```

- We can do the reverse: go from high to low

Using break

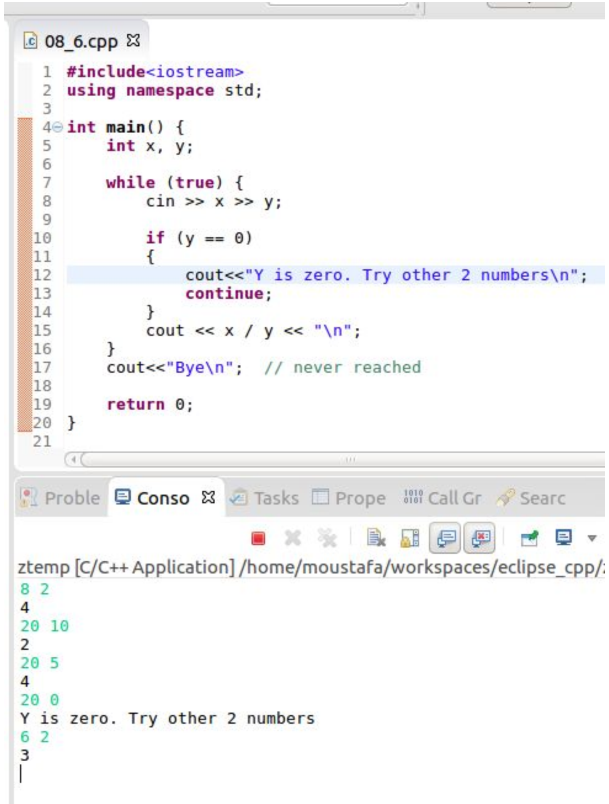


```cpp
#include<iostream>
using namespace std;

int main() {
    int x, y;

    while (true) {
        cin >> x >> y;

        if (y == 0)
        {
            cout<<"Y is zero!!\n";
            break;
        }
        cout << x / y << "\n";
    }
    cout<<"Bye\n";

    return 0;
}

```

```
<terminated> ztemp [C/C++ Application] /home/
8 2
4
20 10
2
20 5
4
20 0
Y is zero!!
Bye
```

- Write a program that KEEPs reading 2 integers and print their division
- If the 2nd number is zero
    - Print Bye
    - End the program
- We can use word break to stop the while and go AFTER it

Using continue



```
#include<iostream>
using namespace std;

int main() {
    int x, y;

    while (true) {
        cin >> x >> y;

        if (y == 0)
        {
            cout<<"Y is zero. Try other 2 numbers\n";
            continue;
        }
        cout << x / y << "\n";
    }
    cout<<"Bye\n";  // never reached

    return 0;
}
```

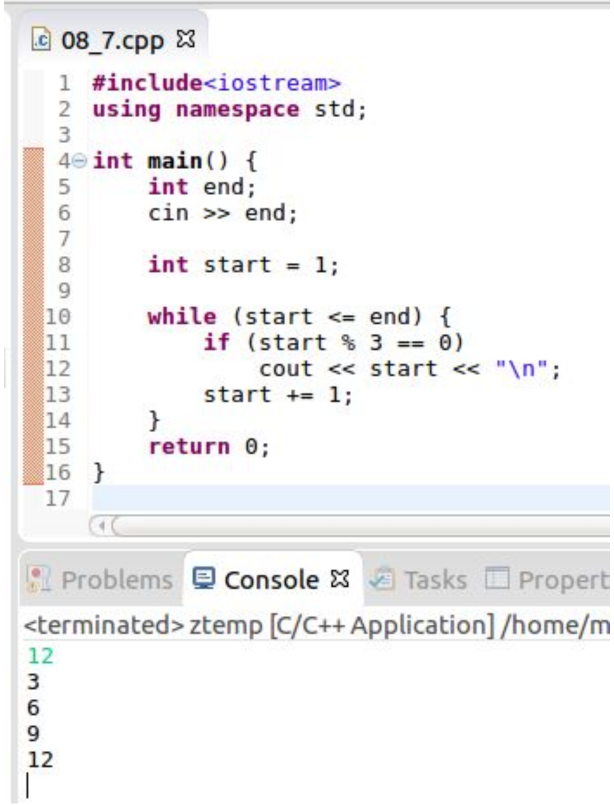
```
ztemp [C/C++ Application] /home/moustafa/workspaces/eclipse_cpp/:
8 2
4
20 10
2
20 5
4
20 0
Y is zero. Try other 2 numbers
6 2
3
```

- Continue tells computer to JUMP to the while start again and continue from there
  - So statements after it INSIDE the while body is skipped

Let’s practice

- When we start practising, you may not understand all examples
- This is ok, keep going
- Syntax is always little, but we practice more

Practice: Numbers divisible by 3

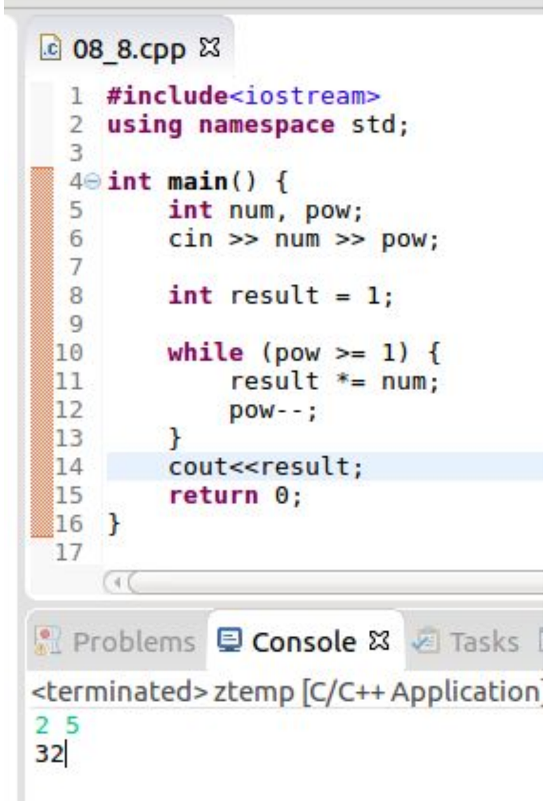


```cpp
#include<iostream>
using namespace std;

int main() {
    int end;
    cin >> end;

    int start = 1;

    while (start <= end) {
        if (start % 3 == 0)
            cout << start << "\n";
        start += 1;
    }
    return 0;
}
```

```
<terminated> ztemp [C/C++ Application] /home/m
12
3
6
9
12
```

- Read an integer X, find all numbers divisible by 3 from 1 to X.
    - These are 3, 6, 9, 12, 15, 18, ….. (multiple of 3)

Practice: Power Function

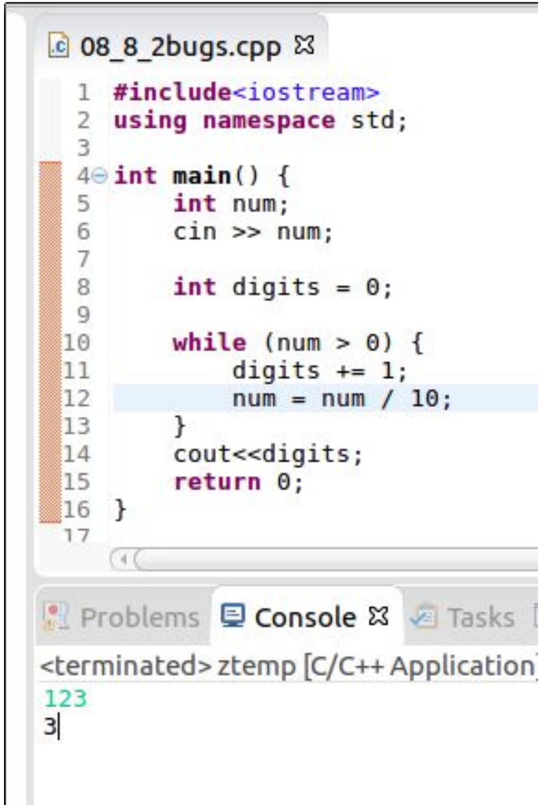


```
#include<iostream>
using namespace std;

int main() {
    int num, pow;
    cin >> num >> pow;

    int result = 1;

    while (pow >= 1) {
        result *= num;
        pow--;
    }
    cout<<result;
    return 0;
}

```

```
<terminated> ztemp [C/C++ Application
2 5
32
```

- Read 2 integers X and Y and compute X^Y.
  - This means X * X * X ….. Y times
  - E.g = 2^5= 2 * 2 * 2 * 2 * 2

Practice: Number of digits

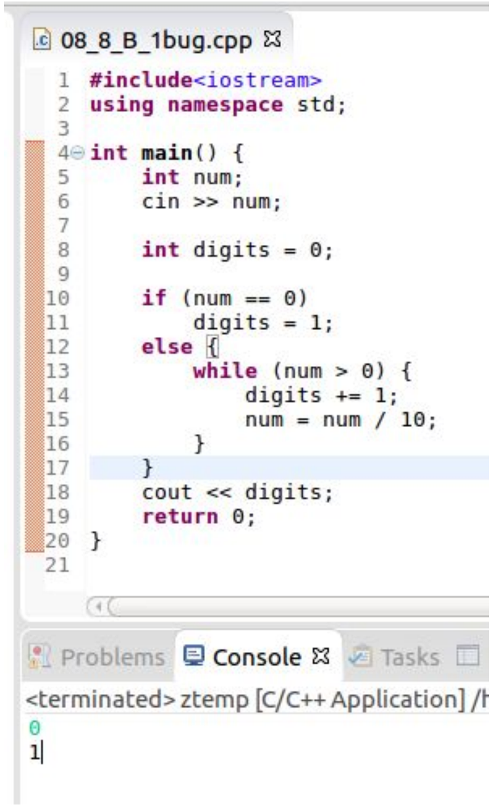


```cpp
#include<iostream>
using namespace std;

int main() {
    int num;
    cin >> num;

    int digits = 0;

    while (num > 0) {
        digits += 1;
        num = num / 10;
    }
    cout<<digits;
    return 0;
}
```

```
<terminated> ztemp [C/C++ Application
123
3
```

- Read a C++ integer and count its number of digits
- There are 2 bugs in this code
    - Find 2 test cases to find them!

Practice: Number of digits - Fixing bugs!

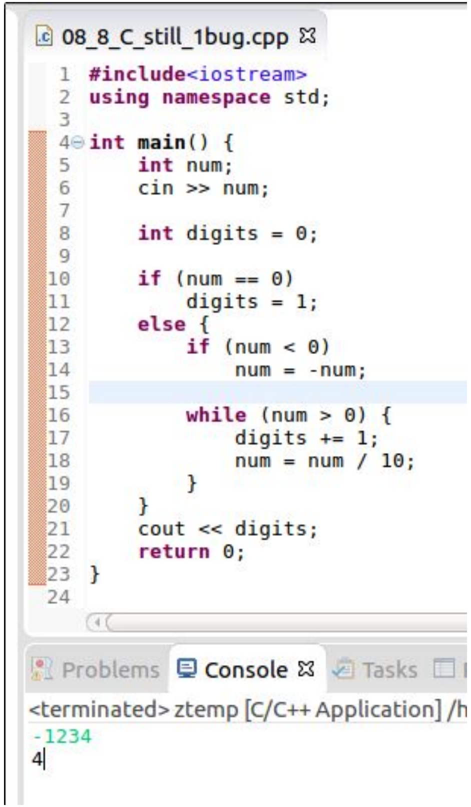


```cpp
#include<iostream>
using namespace std;

int main() {
    int num;
    cin >> num;

    int digits = 0;

    if (num == 0)
        digits = 1;
    else {
        while (num > 0) {
            digits += 1;
            num = num / 10;
        }
    }
    cout << digits;
    return 0;
}
```

```
<terminated> ztemp [C/C++ Application] /h
0
1
```

- Our first bug is: the previous code fails for input 0
    - ○ The loop won’t be accessed as num > 0 condition
- Solution:
    - ○ Special if condition for this special case
- Other bug?

Practice: Number of digits - Fixing bugs!

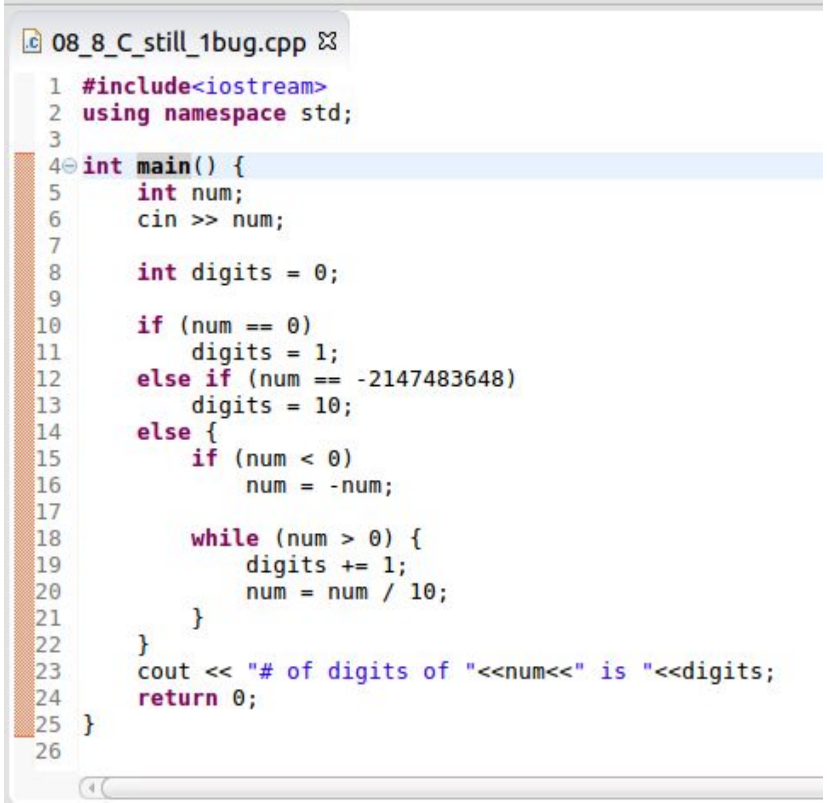


```
#include<iostream>
using namespace std;

int main() {
    int num;
    cin >> num;

    int digits = 0;

    if (num == 0)
        digits = 1;
    else {
        if (num < 0)
            num = -num;

        while (num > 0) {
            digits += 1;
            num = num / 10;
        }
    }
    cout << digits;
    return 0;
}
```

```
<terminated> ztemp [C/C++ Application] /h
-1234
4
```

- Our previous code will fail for negative numbers
- E.g. if we feed -123, the condition fails!
- Simple trick: if it is negative, multiply by -1
- Works well!
- Except a single tricky case. What is it?

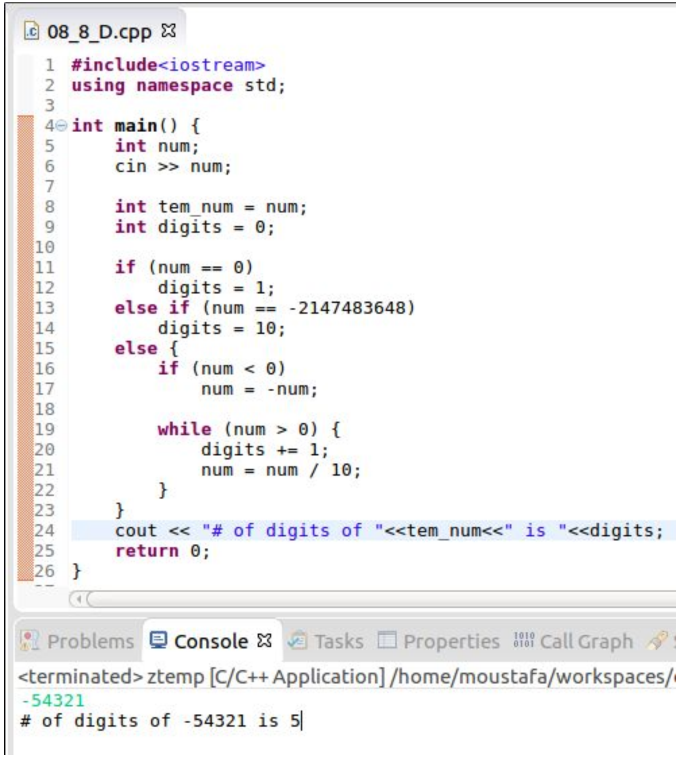
Practice: Number of digits - Fixing bugs!



```
1  #include<iostream>
2  using namespace std;
3
4  int main() {
5      int num;
6      cin >> num;
7
8      int digits = 0;
9
10     if (num == 0)
11         digits = 1;
12     else if (num == -2147483648)
13         digits = 10;
14     else {
15         if (num < 0)
16             num = -num;
17
18         while (num > 0) {
19             digits += 1;
20             num = num / 10;
21         }
22     }
23     cout << "# of digits of "<<num<<" is "<<digits;
24     return 0;
25 }
26
```

- How could -num be wrong?
- We said last time integer limits are:
  - -2147483648 to 2147483647
- If we did -num on the max number it become: 2147483648
  - But this is > 2147483647 ⇒ Overflow
  - Solution: Special case handling
- We wanted to change output message as line 23. What is wrong?

Practice: Number of digits

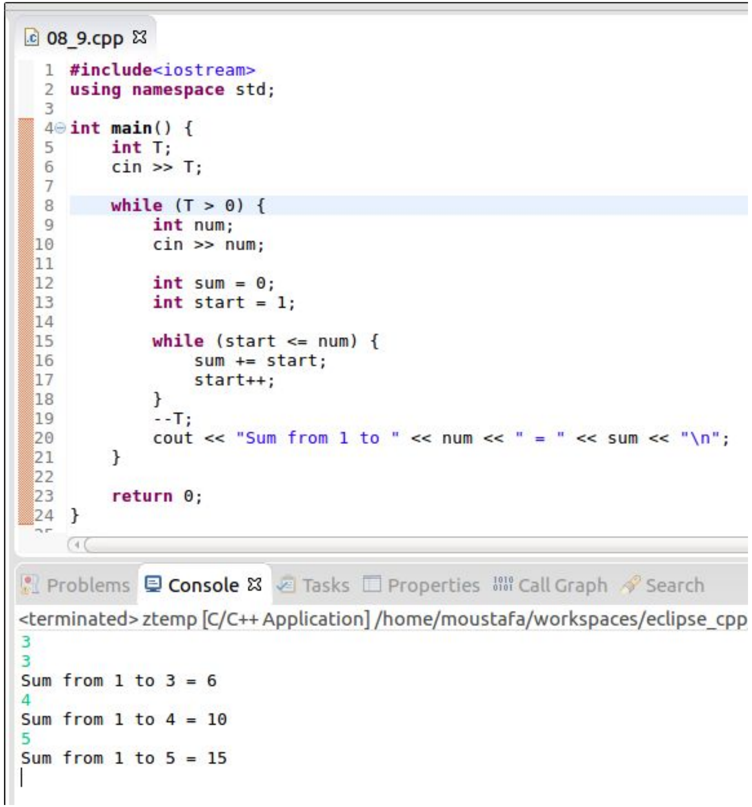


```cpp
#include<iostream>
using namespace std;

int main() {
    int num;
    cin >> num;

    int tem_num = num;
    int digits = 0;

    if (num == 0)
        digits = 1;
    else if (num == -2147483648)
        digits = 10;
    else {
        if (num < 0)
            num = -num;

        while (num > 0) {
            digits += 1;
            num = num / 10;
        }
    }
    cout << "# of digits of "<<tem_num<<" is "<<digits;
    return 0;
}
```

```
<terminated> ztemp [C/C++ Application] /home/moustafa/workspaces/
-54321
# of digits of -54321 is 5
```

- Num, the input, was divided till be zero. So we lost its **original** value!
- Solution: Make a copy
- **Lesson**
  - It takes time to be a strong programmer
    - Clean readable code
    - Short code
    - Well tested code

Nested loop



```cpp
#include<iostream>
using namespace std;

int main() {
    int T;
    cin >> T;

    while (T > 0) {
        int num;
        cin >> num;

        int sum = 0;
        int start = 1;

        while (start <= num) {
            sum += start;
            start++;
        }
        --T;
        cout << "Sum from 1 to " << num << " = " << sum << "\n";
    }

    return 0;
}
```

```
<terminated> ztemp [C/C++ Application] /home/moustafa/workspaces/eclipse_cpp
3
3
Sum from 1 to 3 = 6
4
Sum from 1 to 4 = 10
5
Sum from 1 to 5 = 15
```

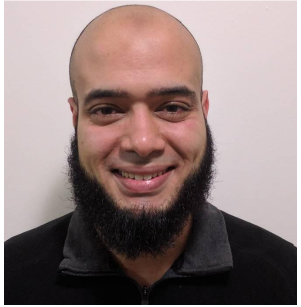
- Write a program that reads integer T for T test cases.
- Then read T numbers: for each number N **print sum of 1 to N**
- Remember, we can replace the sum with formula N * (N+1) / 2
  - ○ Which is more efficient?

Programming4kids

While Loops

Mostafa Saad Ibrahim
Computer Vision Researcher @ Huawei Canada
PhD - Simon Fraser University
Bachelor / Msc - FCI Cairo University

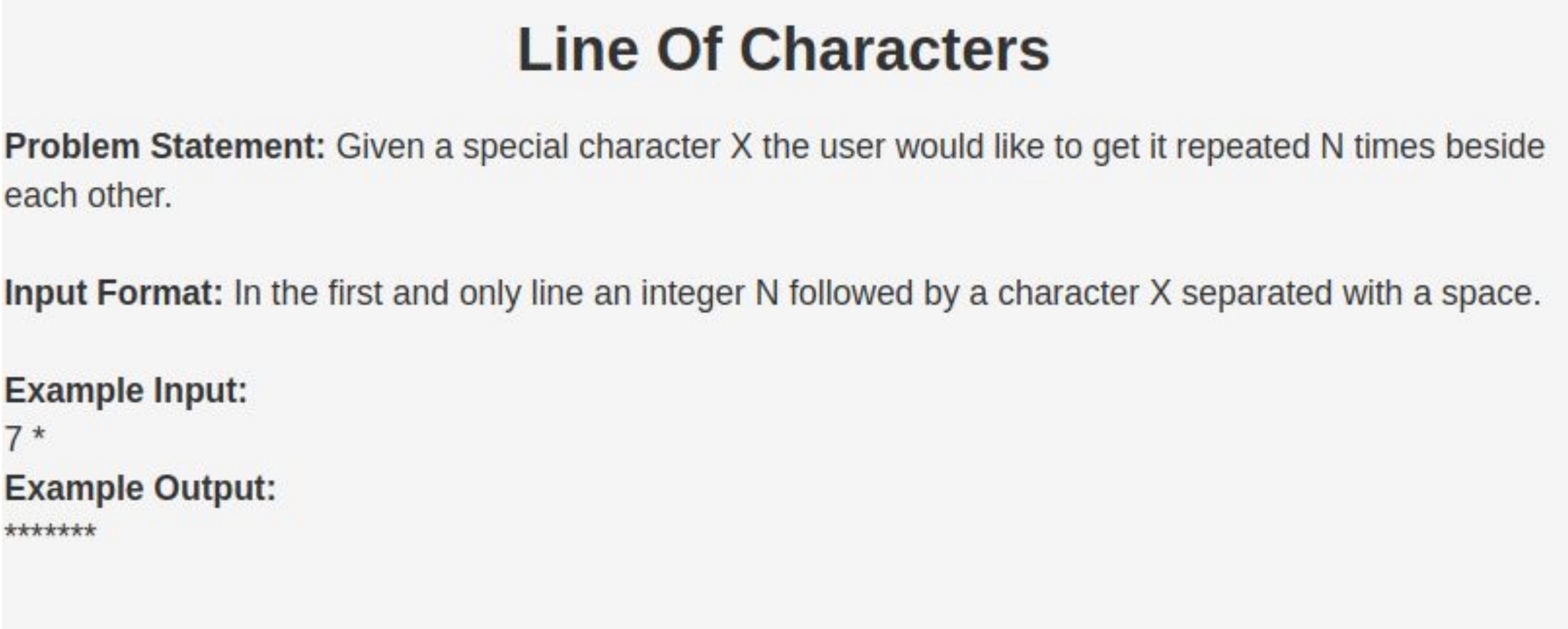
Ex-(Software Engineer / Teaching Assistant)



Homework 1: Print Range

- Given a starting number X and an ending number Y, print all numbers between X and Y inclusive, each on a line.
- Input 3 7
- Output
  - 3
  - 4
  - 5
  - 6
  - 7

Homework 2: Line Of Characters



Line Of Characters

Problem Statement: Given a special character X the user would like to get it repeated N times beside each other.

Input Format: In the first and only line an integer N followed by a character X separated with a space.

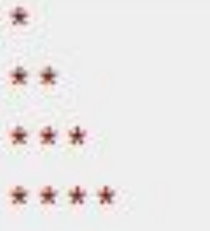
Example Input:

```
7 *
```

Example Output:

```
*******
```

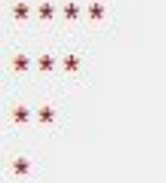
Homework 3: Print left angled triangle

- Given a number N. Print a left angled triangle that has N rows.
- Input 4
- Output
- 

```
*
* *
* * *
* * * *
```

Homework 4: Print face down left angled triangle

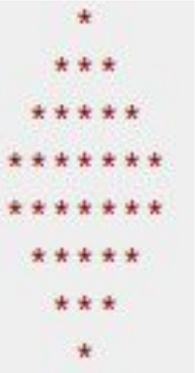
- Given a number N. Print a face down left angled triangle that has N rows.
- Input 4
- Output
-



```
* * * *
* * *
* *
*
```

Homework 5: Print diamond

- Given a number N. Print diamond of 2N rows as below.
- Input 4
- Output
-



```
      *
    * * *
  * * * * *
* * * * * * *
* * * * * * *
  * * * * *
    * * *
      *
```

Homework 6: Special Average

- Read integer N, followed by reading N numbers. Print 2 values
  - The average of the numbers in odd positions (1st, 3rd, 5th, …)
  - The average of the numbers in even positions (2nd, 4th, 6th, …)
- Input
  - 6 10 100 20 200 30 600
- Output
  - 20 300
- Explantation
  - (10+20+30)/3 = 20
  - (100+200+600)/3 = 300

Homework 7: Special multiples 1

- Read an integer N (1 <= 200): print all numbers that satisfy the following property
  - Either number is divisible by 8
  - Or divisible by both 4 and 3
- Input: 100
- Output: 0 8 12 16 24 32 36 40 48 56 60 64 72 80 84 88 96

Homework 8: Special multiples 2

- Read an integer N (1 <= 30): Print the first N numbers that are
    - multiple of 3 but not multiple of 4
- Input: 11
- Output: 3 6 9 15 18 21 27 30 33 39 42
- Notice
    - 12 is divisible by both 3 and 4 ⇒ so excluded

Homework 9: Find NO

- Read integer N, then read N strings.
    - Print only the strings (of 2 letters). These 2 letters must be letter ‘N’ and letter ‘O’ (regardless of lower/upper case/order)
    - E.g. print “No”, “ON”, “no”   but ignore e.g. “YEs”, “Noooo”
    - That is, the word of 2 letters only N and O
- Input
    - 9 Yss NO noOO oN Mostafa no nN oOOooo oO
- Output
    - NO oN no

Homework 10: Reverse number

- Read an integer N, then find its reverse integer R
  - Print R R*3
- input ⇒ Output
  - 123 ⇒ 321 963

Homework 11: Minimum of values

- Read T for number of test cases. For each test case read integer N: number of integers. For each test case, print the minimum of the N integers.
- Input
    - 2   6   10 50 20 70 30 4   3 10 5 30
    - Notice here we have 2 test cases
        - 6   10 50 20 70 30 4   [6 numbers to read]
        - 3   10 5 30
- Output
    - 4
    - 5

Homework 12: Multiplication table

- Read an integer N and M, then print NxM lines for their multiplication table.
- Input 3 4
- Output
    - 1 x 1 = 1
    - 1 x 2 = 2
    - 1 x 3 = 3
    - 1 x 4 = 4
    - 2 x 1 = 2
    - 2 x 2 = 4
    - 2 x 3 = 6
    - 2 x 4 = 8
    - 3 x 1 = 3
    - 3 x 2 = 6
    - 3 x 3 = 9
    - 3 x 4 = 12

Homework 13: Special Sum

- Read T for number of test cases. For each test case read integer N. Then read N integers a, b, c, ….. and compute the **sum** of:
  - (a, b*b, c*c*c, d*d*d*d, e*e*e*e*e……)
  - That is the k-th number is repeated k times
- Input:
  - 2
  - 3   5 7 2
  - 4 1 2 3 4
- Output
  - 62 [as (5 + 7*7 + 2*2*2) = 62]
  - 288 [as (1+2*2+3*3*3+4*4*4*4) = 288]

تم بحمد الله

علمكم الله ما ينفعكم

ونفعكم بما تعلمتم

وزادكم علماً