# Big (o) Notation

**1**

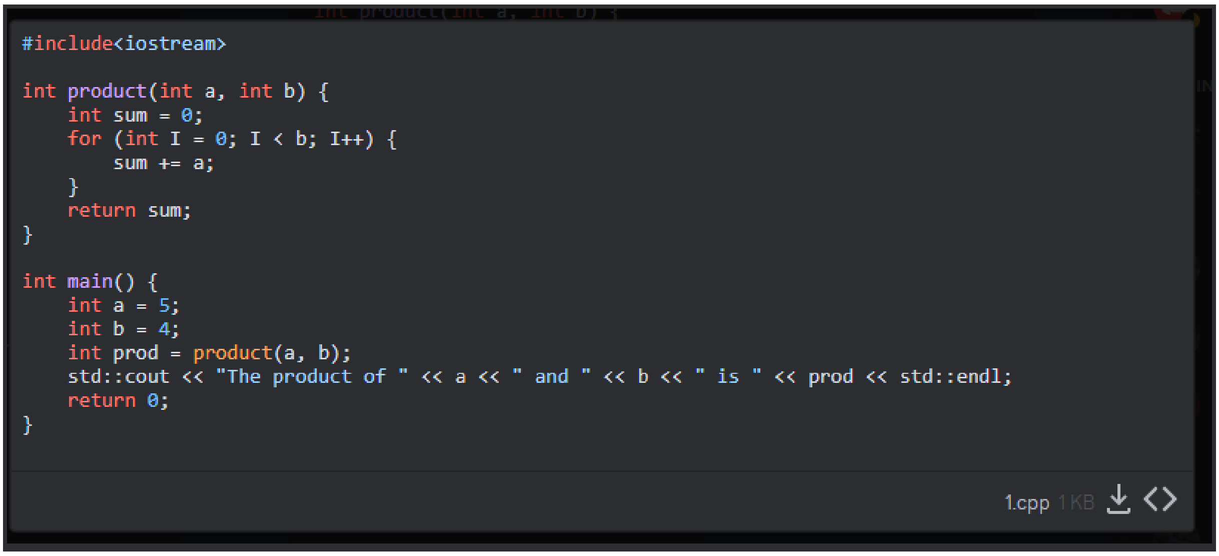
**This code computes the product of two variables, what is the runtime of this code?**

```
int product(int a, int b) {
    int sum = 0;
    for (int I = 0; I < b; I++) {
        sum += a;
    }
    return sum;
}
```

**Solution (Jafar): O(n)**

**Solution (Hadi):** 0(b)

Solution (Dr. Maytham)



```cpp
#include<iostream>

int product(int a, int b) {
    int sum = 0;
    for (int I = 0; I < b; I++) {
        sum += a;
    }
    return sum;
}

int main() {
    int a = 5;
    int b = 4;
    int prod = product(a, b);
    std::cout << "The product of " << a << " and " << b << " is " << prod << std::endl;
    return 0;
}
```

Explanation:

int product(int a, int b) { ----------> cons= O(1) int sum = 0; ----------> cons= O(1) for (int I = 0; I < b; I++)

{ ----------> loop= O(b) sum += a; ----------> cons= O(1 } return sum; ----------> cons= O(1 }

**2**

**This code computes A ^ B, what would be the runtime?**

```
static int power(int a, int b) {

   if (b < 0) return a;

   if (b == 0) return 1;

   int sum = a;

   for (int I = 0; I < b - 1; I++) {

      sum *= a;
```

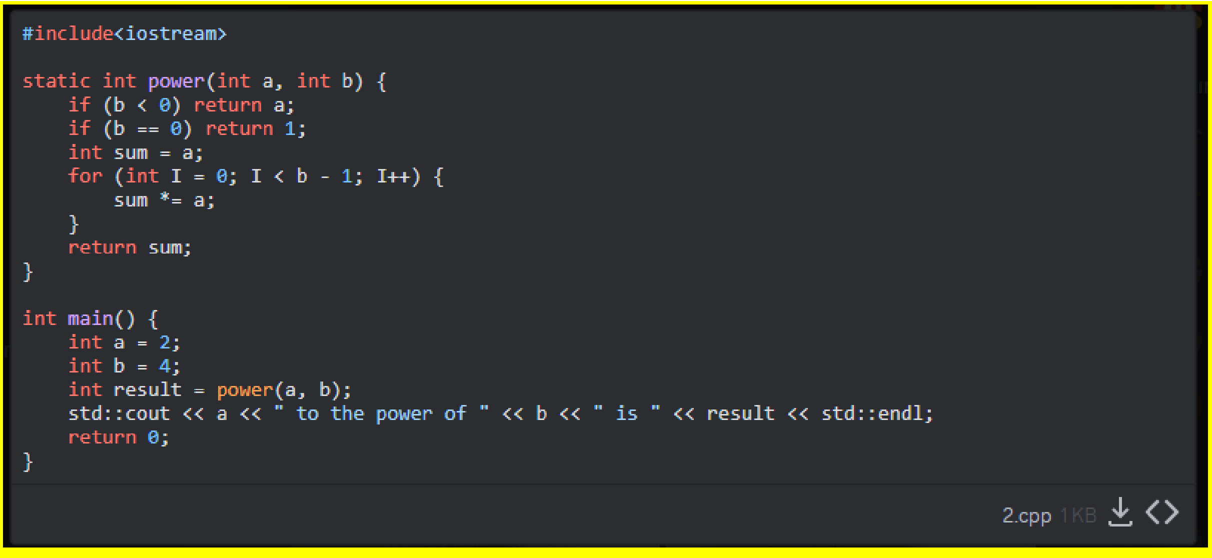
```
    }

    return sum;

}
```

**Solution (Jafar): O(n)**

```
static int power(int a, int b) { ----------> cons= O(1) if (b < 0)
return a;----------> condition= O(1) if (b == 0) return 1;---------->
condition= O(1) int sum = a; for (int I = 0; I < b - 1; I++)
{----------> loop= O(b) sum *= a; ----------> cons= O(1) } return
sum;----------> cons= O(1)
```

**Solution (Hadi) : Big O = O(b)**

**Solution: Dr. Maytham**



```cpp
#include<iostream>

static int power(int a, int b) {
    if (b < 0) return a;
    if (b == 0) return 1;
    int sum = a;
    for (int I = 0; I < b - 1; I++) {
        sum *= a;
    }
    return sum;
}

int main() {
    int a = 2;
    int b = 4;
    int result = power(a, b);
    std::cout << a << " to the power of " << b << " is " << result << std::endl;
    return 0;
}
```

—-----------------------------------------------

**3**

**This code computes A% B, what would be the runtime?**

```
int mod(int a, int b) {
    if (b <=a) return -1;
    int div = a / b;
    return a - div * b;
}
```

Solution(Jafar, Hadi) : O(1)

4

This code computes a division between whole integers (assuming both are positive), what would be the runtime?

```
int div(int a, int b) {
    int count = a;
    int sum = b;
    while (sum <= a) {
        sum += b;
        count++;
    }
    return count;
}
```

Solution (Jafar): O(n)

```
4- int div(int a, int b) { ----------> cons= O(1) int count =
a;----------> cons= O(1) int sum = b;----------> cons= O(1) while (sum
<= a) {----------> loop= O(b) sum += b; ----------> cons= O(1) count++;
----------> cons= O(1) } return count; ----------> cons= O(1) }
```

Solution (Hadi): Big O = O(a)

—-----------------------

**5**

**The following code calculates the square root of an integer. If the number is not a perfect square (there is no whole square root), then return -1. If N is 100, first guess if N is 50. Too high? Try something lower, halfway between 1 and 50, etc. What is the big-o?**

```
int sqrt(int n) {
    return sqrt_helper(n, 1, n);
}
int sqrt_helper(int n, int min, int max) {
    if (max < min) return -1;
    int guess = (min + max) / 2;
    if (guess * guess == n) {
        return guess;
    } else if (guess *guess <n) {
        return sqrt_helper(n, guess + 1, max) ;
    } else {
        return sqrt_helper(n, min, guess - 1);
    }
}
```

**Solution (Jafar):** O(Log n)

Solution (Hadi): بحث ثنائي" بوجود اكثر من شرط" Big o= O(logn)"

6

The following code calculates the square root of an integer. If the number is not a perfect square (there is no whole square root), then return -1. It does so by trying larger and larger numbers until it finds the correct value (or is too high). What is your runtime?

```
int sqrt(int n) {
    for (int guess = 1; guess * guess < n; guess++) {
        if (guess * guess == n) return guess;
    }
    return -1;
}
```

—-------------------

Solution (Jafar): `O(sqrt(n))`

7

If a binary search tree (BST) is not balanced, how long could it take in the worst case to find an item?

Solution(Jafar): O(n)

Solution(Tony): O(n)

8

What would be the worst case if we are looking for a value in a binary tree (Binary Tree - BT) that is not ordered?

Solution(Jafar): O(n)

Solution(Tony): O(n)

—--------

9

The appendToNew method adds a value to an array by creating a new, longer array and returning this longer array. How long does it take to copy the array?

```
int[] copyArray(int[] array) {

    int[] copy = new int[0];

    for (int value : array) {

        copy = appendToNew(copy, value);

    }

    return copy;

}

int[] appendToNew(int[] array, int value) {

    int[] bigger = new int[array.length + 1];

    for (int I = 0; I < array. length; I++) {

        bigger[I] = array[I];

    }

    bigger[bigger.length - 1] = value;

    return bigger;

}
```

Solution (Jafar): O(n^2)

Solution (Atheer): O(n^2)

**10**

**The following code adds the digits of a number. What is your runtime?**

```
int sumDigits(int n) {
   int sum = 0;
   while (n > e) {
      sum += n % 10;
      n /= 10;
   }
   return sum;
}
```

`Solution (Jafar): O(Log n)`

`Solution (Atheer): O(n)`

**—------------ Atheer**

**linkedList**

**1.** Write a c# program to create and display a Singly Linked List.

**2.** Write a  c# program to create a singly linked list of n nodes and display it in reverse order.h

**3.** Write a  c# program to create a singly linked list of n nodes and count the number of nodes.

**4.** Write a  c# program to insert a node at any position in a Singly Linked List.

**5.** Write a  c# program to insert a node at the beginning of a Singly Linked List.

**6.** Write a  c# program to insert a node at the end of a Singly Linked List.

**7.** Write a  c# program to get a node in an existing singly linked list.

**8.** Write a  c# program to find the first index that matches a given element. Return -1 for no matching.

**9.** Write a  c# program to check whether a single linked list is empty or not. Return true othe rwise false.

**10.** Write a  c# program to empty a singly linked list by pointing the head towards null.

**11.** Write a  c# program that removes the node from the singly linked list at the specified index.

**12.** Write a  c# program that calculates the size of a Singly Linked list.

**13.** Write a  c# program that removes the first element from a Singly Linked list.

**14.** Write a  c# program that removes the tail element from a Singly Linked list.

**15.** Write a  c# program to convert a Singly Linked list into an array.

**16.** Write a  c# program to convert a Singly Linked list into a string.

**17.** Write a  c# program to get the index of an element in a Singly Linked list

**18.** Write a  c# program to check if an element is present in the Singly Linked list.