Name of the task : Climbing Stair

Number of the task : 1

Team Number: 1

ID	Name
20210897	مروان علي أحمد عبد الله
20210914	مريم سعد محمد سعد
20210917	مريم علي عبد الرحمن علي
20210923	مريم محمد حامد عبد الحميد شلبي (team leader)
20210962	منة الله محمد حفني محمد
20211063	يوسف اسماعيل رياض ابراهيم

first algorithm

Pseudo code :

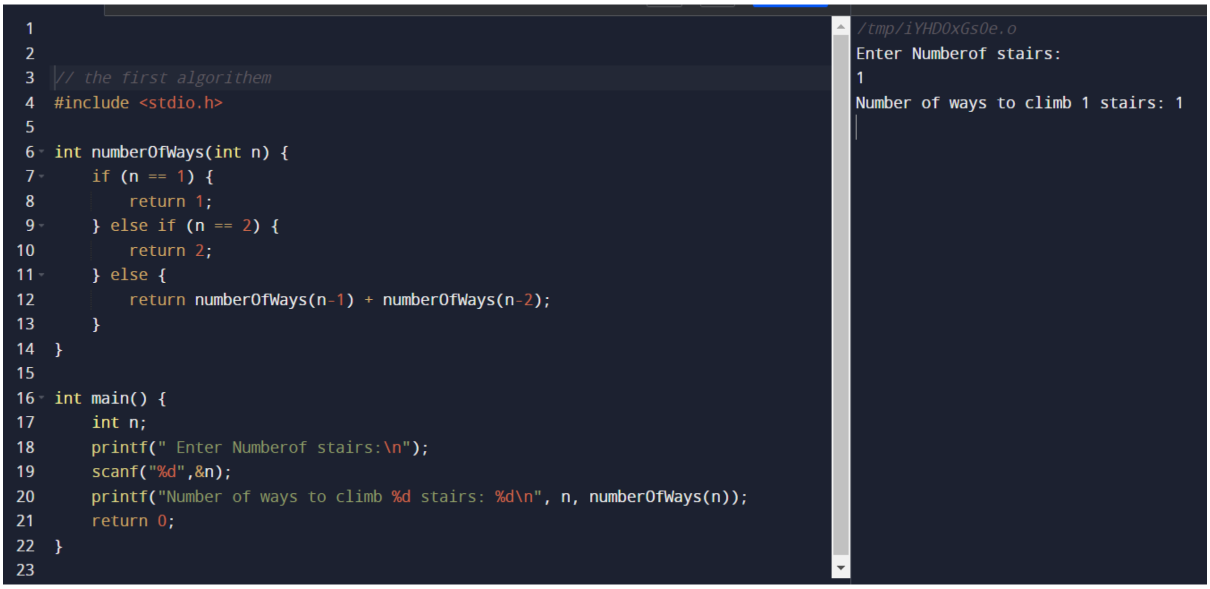
```
int climbStairs(int n)
        if(n == 0 || n == 1)
                return 1;
        else
                Create array numberOfWays[n + 1];
                Initialize numberOfWays[0] = 1 and numberOfWays[1] = 1
        for i <- 2 to i <= n
        Do numberOfWays[i] <- numberOfWays[i - 1] + numberOfWays[i - 2];
return numberOfWays[n];
```

Complexity time:

$$\sum_{i=2}^{n} 1 = \text{n-2+1} \approx O(n)$$

Best case	Average case	Worst case
$\Omega(1)$	$\Theta(n)$	$O(n)$
When number of steps = 0 or 1	When number of steps > 1 to n	When number of steps > 1 to n

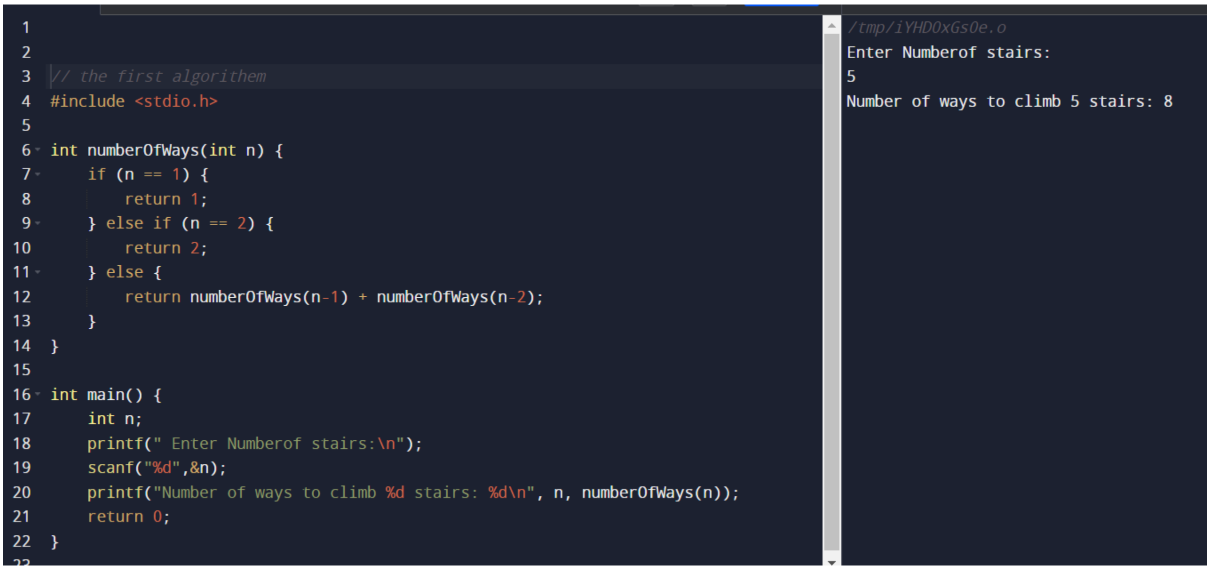
Best case:



```c
// the first algorithem
#include <stdio.h>

int numberOfWays(int n) {
    if (n == 1) {
        return 1;
    } else if (n == 2) {
        return 2;
    } else {
        return numberOfWays(n-1) + numberOfWays(n-2);
    }
}

int main() {
    int n;
    printf(" Enter Numberof stairs:\n");
    scanf("%d",&n);
    printf("Number of ways to climb %d stairs: %d\n", n, numberOfWays(n));
    return 0;
}
```

```
/tmp/iYHDOxGs0e.o
Enter Numberof stairs:
1
Number of ways to climb 1 stairs: 1
```

Average case:



```c
// the first algorithem
#include <stdio.h>

int numberOfWays(int n) {
    if (n == 1) {
        return 1;
    } else if (n == 2) {
        return 2;
    } else {
        return numberOfWays(n-1) + numberOfWays(n-2);
    }
}

int main() {
    int n;
    printf(" Enter Numberof stairs:\n");
    scanf("%d",&n);
    printf("Number of ways to climb %d stairs: %d\n", n, numberOfWays(n));
    return 0;
}
```

```
/tmp/iYHDOxGs0e.o
Enter Numberof stairs:
5
Number of ways to climb 5 stairs: 8
```

Second algorithm

Pseudo code :

```
int numberOfWays(int n) {
          /*two base cases n==1 n==2*/
                   if (n == 1) {
                            return 1;}
                  else if (n == 2){
                           return 2; }
                  else {
                           return numberOfWays(n-1) + numberOfWays(n-2); }
}
```

Complexity time:

$T(n) = t(n-1) + t(n-2)$ **$T(1) = 1, T(2) = 2$**

$(n-1) > (n-2)$ **then** **$T(n) = 2T(n-2) + b$**

$= 2[T(n-3) + T(n-4) + b] + b$

by substituting $T(n-2)$ in (2)

$2[T(n-4) + T(n-4) + b] + b$

$= 2^2T(n-4) + 2b + b$

$= 2^2[T(n-5) + T(n-6) + b] + 2b + b$

by substituting $T(n-4)$ in (2)

$\geq 2^3T(n-6) + (2^2 + 2^1 + 2^0)b. . .$

$2^kT(n-2k) + (2^{k-1} + 2^{k-2} + . . . + 2^1 + 2^0)b$

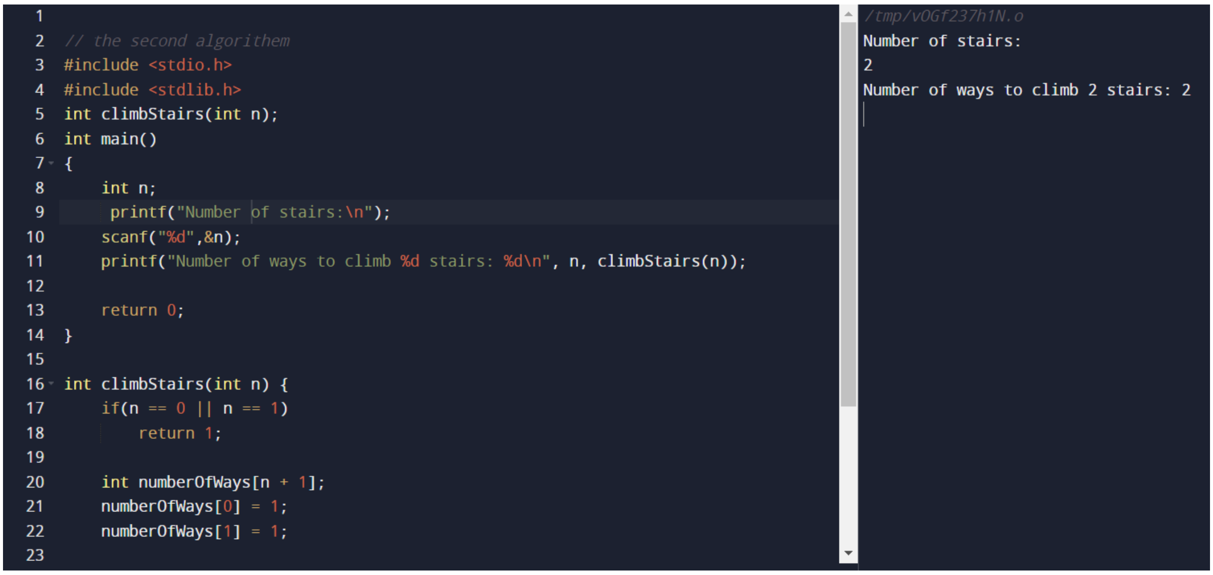
$= 2^kT(n-2k) + (2^k - 1)b$

Hence $T(n) \geq 2\ (n-2) / 2\ T(2) + [2\ (n-2) / 2 - 1]b$

$= (b + c)2^{(n-2)/2} - b$

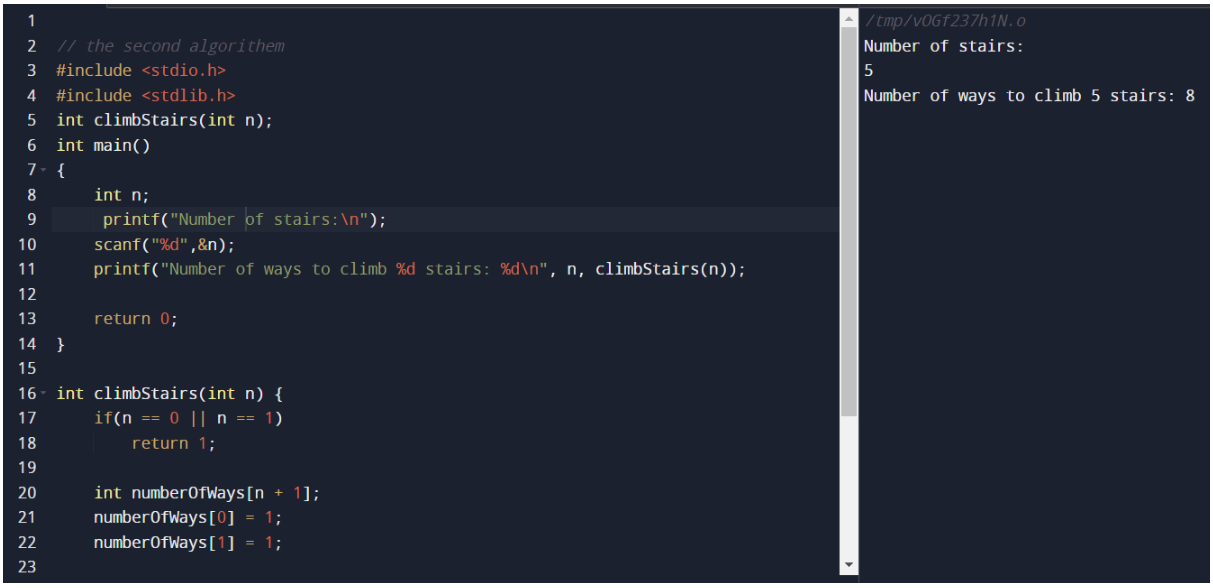
$$= [(b + c) / 2]*(2)^{n/2} – b \approx \boldsymbol{O(2^n)}$$

Best case	Average case	Worst case
$\Omega(1)$	$\Theta(2^n)$	$O(2^n)$
When number of steps = 1 or 2	When number of steps > 2	When number of steps > 2



```

// the second algorithem
#include <stdio.h>
#include <stdlib.h>
int climbStairs(int n);
int main()
{
    int n;
     printf("Number of stairs:\n");
    scanf("%d",&n);
    printf("Number of ways to climb %d stairs: %d\n", n, climbStairs(n));

    return 0;
}

int climbStairs(int n) {
    if(n == 0 || n == 1)
        return 1;

    int numberOfWays[n + 1];
    numberOfWays[0] = 1;
    numberOfWays[1] = 1;

```

```
/tmp/vOGf237h1N.o
Number of stairs:
2
Number of ways to climb 2 stairs: 2
```



```

// the second algorithem
#include <stdio.h>
#include <stdlib.h>
int climbStairs(int n);
int main()
{
    int n;
     printf("Number of stairs:\n");
    scanf("%d",&n);
    printf("Number of ways to climb %d stairs: %d\n", n, climbStairs(n));

    return 0;
}

int climbStairs(int n) {
    if(n == 0 || n == 1)
        return 1;

    int numberOfWays[n + 1];
    numberOfWays[0] = 1;
    numberOfWays[1] = 1;

```

```
/tmp/vOGf237h1N.o
Number of stairs:
5
Number of ways to climb 5 stairs: 8
```

Comparison between algorithm one and two:

	Algorithm one	Algorithm two
Best case	$\Omega(1)$	$\Omega(1)$
Average case	$\Theta(\mathrm{n})$	$\Theta(2^n)$
Worst case	$\mathrm{O}(n)$	$O(2^n)$

The best algorithm:

is the first algorithm in average case and worst case.