COMP1009
Data Structures and Algorithms

Theory problems exercise sheet: worked solutions

Dr Julian Rathke

23rd April 2008

Question One

1 Consider the program (valid for inputs $n \geq 1$)

```
foo(int n) {
   bar();
   if (n==1)
      return;
   foo(n-1);
   foo(n-1);
   foo(n-1);
}
```

(a) Let $C(n)$ be the number of times the function `bar()` is called when we call `foo(n)`. Write down a recurrence relation for $C(n)$. *(2 marks)*

$C(n) =$

(b) Write down the boundary condition for the recurrence relation. *(1 marks)*

$C(1) =$

(c) Using the recurrence relation to compute $C(2)$, $C(3)$ and $C(4)$. *(3 marks)*

$C(2) =$

$C(3) =$

$C(4) =$

(d) Prove by induction that $f(n) = \frac{3^n - 1}{2}$ satisfies the recurrence relation for $C(n)$. *(5 marks)*

1 Consider the program (valid for inputs $n \geq 1$)

```
foo(int n) {
   bar();
   if (n==1)
      return;
   foo(n-1);
   foo(n-1);
   foo(n-1);
}
```

(a) Let $C(n)$ be the number of times the function `bar()` is called when we call `foo(n)`. Write down a recurrence relation for $C(n)$. *(2 marks)*

$C(n) =$

(b) Write down the boundary condition for the recurrence relation. *(1 marks)*

$C(1) =$

(c) Using the recurrence relation to compute $C(2)$, $C(3)$ and $C(4)$. *(3 marks)*

$C(2) =$

$C(3) =$

$C(4) =$

(d) Prove by induction that $f(n) = \frac{3^n - 1}{2}$ satisfies the recurrence relation for $C(n)$. *(5 marks)*

Question One (a)

How many times is `bar()` called when `foo(n)` is evaluated?

C(n) =

```
foo(int n) {
    bar();
    if (n==1)
      return;
    foo(n-1);
    foo(n-1);
    foo(n-1);
}
```

Question One (a)

How many times is `bar()` called when `foo(n)` is evaluated?

$C(n) =$

```
foo(int n) {
    bar();
    if (n==1)
      return;
    foo(n-1);
    foo(n-1);
    foo(n-1);
}
```

Question One (a)

How many times is `bar()` called when `foo(n)` is evaluated?

$C(n) = \boxed{1}$

```
foo(int n) {
    bar();
    if (n==1)
       return;
    foo(n-1);
    foo(n-1);
    foo(n-1);
}
```

Question One (a)

How many times is `bar()` called when `foo(n)` is evaluated?

$C(n) = 1$

```
foo(int n) {
    bar();
    if (n==1)
      return;
    foo(n-1);
    foo(n-1);
    foo(n-1);
}
```

Question One (a)

How many times is `bar()` called when `foo(n)` is evaluated?

$C(n) = 1$

How many times is `bar()` called when `foo(n-1)` is evaluated?

```
foo(int n) {
    bar();
    if (n==1)
      return;
    foo(n-1);
    foo(n-1);
    foo(n-1);
}
```

Question One (a)

How many times is `bar()` called when `foo(n)` is evaluated?

$$C(n) = 1 + C(n-1)$$

How many times is `bar()` called when `foo(n-1)` is evaluated?

```
foo(int n) {
    bar();
    if (n==1)
        return;
    foo(n-1);
    foo(n-1);
    foo(n-1);
}
```

Question One (a)

How many times is `bar()` called when `foo(n)` is evaluated?

$$C(n) = 1 + C(n\text{-}1) + C(n\text{-}1)$$

How many times is `bar()` called when `foo(n-1)` is evaluated?

```
foo(int n) {
    bar();
    if (n==1)
        return;
    foo(n-1);
    foo(n-1);
    foo(n-1);
}
```

Question One (a)

How many times is `bar()` called when `foo(n)` is evaluated?

$$C(n) = 1 + C(n-1) + C(n-1) + \boxed{C(n-1)}$$

How many times is `bar()` called when `foo(n-1)` is evaluated?

```
foo(int n) {
    bar();
    if (n==1)
       return;
    foo(n-1);
    foo(n-1);
    foo(n-1);
}
```

Question One (a)

How many times is `bar()` called when `foo(n)` is evaluated?

$$C(n) = 1 + C(n-1) + C(n-1) + C(n-1)$$

$$= 3C(n-1) + 1$$

How many times is `bar()` called when `foo(n-1)` is evaluated?

```
foo(int n) {
    bar();
    if (n==1)
       return;
    foo(n-1);
    foo(n-1);
    foo(n-1);
}
```

1 Consider the program (valid for inputs $n \geq 1$)

```
foo(int n) {
   bar();
   if (n==1)
      return;
   foo(n-1);
   foo(n-1);
   foo(n-1);
}
```

(a) Let $C(n)$ be the number of times the function `bar()` is called when we call `foo(n)`. Write down a recurrence relation for $C(n)$. *(2 marks)*

$C(n) =$ 3C(n-1) + 1

(b) Write down the boundary condition for the recurrence relation. *(1 marks)*

$C(1) =$

(c) Using the recurrence relation to compute $C(2)$, $C(3)$ and $C(4)$. *(3 marks)*

$C(2) =$

$C(3) =$

$C(4) =$

(d) Prove by induction that $f(n) = \frac{3^n - 1}{2}$ satisfies the recurrence relation for $C(n)$. *(5 marks)*

THEORY PROBLEMS FOR DATA STRUCTURES AND ALGORITHMS (COMP1009)

1 Consider the program (valid for inputs $n \geq 1$)

```
foo(int n) {
   bar();
   if (n==1)
      return;
   foo(n-1);
   foo(n-1);
   foo(n-1);
}
```

(a) Let $C(n)$ be the number of times the function `bar()` is called when we call `foo(n)`. Write down a recurrence relation for $C(n)$. *(2 marks)*

$C(n) =$ 3C(n-1) + 1

(b) Write down the boundary condition for the recurrence relation. *(1 marks)*

$C(1) =$

(c) Using the recurrence relation to compute $C(2)$, $C(3)$ and $C(4)$. *(3 marks)*

$C(2) =$

$C(3) =$

$C(4) =$

(d) Prove by induction that $f(n) = \frac{3^n-1}{2}$ satisfies the recurrence relation for $C(n)$. *(5 marks)*

Question One (b)

How many times is `bar()` called when `foo(1)` is evaluated?

```
foo(int n) {
    bar();
    if (n==1)
      return;
    foo(n-1);
    foo(n-1);
    foo(n-1);
}
```

Question One (b)

How many times is `bar()` called when `foo(1)` is evaluated?

```
foo(int n) {
    bar();
    if (n==1)
      return;
    foo(n-1);
    foo(n-1);
    foo(n-1);
}
```

Question One (b)

How many times is `bar()` called when `foo(1)` is evaluated?

```
foo(int n) {
    bar();
    if (n==1)
      return;
    foo(n-1);
    foo(n-1);
    foo(n-1);
}
```

Question One (b)

How many times is `bar()` called when `foo(1)` is evaluated?

$$C(1) = 1$$

```
foo(int n) {
    bar();
    if (n==1)
      return;
    foo(n-1);
    foo(n-1);
    foo(n-1);
}
```

1 Consider the program (valid for inputs $n \geq 1$)

```
foo(int n) {
   bar();
   if (n==1)
      return;
   foo(n-1);
   foo(n-1);
   foo(n-1);
}
```

(a) Let $C(n)$ be the number of times the function `bar()` is called when we call `foo(n)`. Write down a recurrence relation for $C(n)$. *(2 marks)*

$C(n) =$ 3C(n-1) + 1

(b) Write down the boundary condition for the recurrence relation. *(1 marks)*

$C(1) =$ 1

(c) Using the recurrence relation to compute $C(2)$, $C(3)$ and $C(4)$. *(3 marks)*

$C(2) =$

$C(3) =$

$C(4) =$

(d) Prove by induction that $f(n) = \frac{3^n - 1}{2}$ satisfies the recurrence relation for $C(n)$. *(5 marks)*

1 Consider the program (valid for inputs $n \geq 1$)

```
foo(int n) {
   bar();
   if (n==1)
      return;
   foo(n-1);
   foo(n-1);
   foo(n-1);
}
```

(a) Let $C(n)$ be the number of times the function `bar()` is called when we call `foo(n)`. Write down a recurrence relation for $C(n)$. *(2 marks)*

$C(n) =$ 3C(n-1) + 1

(b) Write down the boundary condition for the recurrence relation. *(1 marks)*

$C(1) =$ 1

(c) Using the recurrence relation to compute $C(2)$, $C(3)$ and $C(4)$. *(3 marks)*

$C(2) =$

$C(3) =$

$C(4) =$

(d) Prove by induction that $f(n) = \frac{3^n - 1}{2}$ satisfies the recurrence relation for $C(n)$. *(5 marks)*

1 Consider the program (valid for inputs $n \geq 1$)

```
foo(int n) {
   bar();
   if (n==1)
      return;
   foo(n-1);
   foo(n-1);
   foo(n-1);
}
```

(a) Let $C(n)$ be the number of times the function `bar()` is called when we call `foo(n)`. Write down a recurrence relation for $C(n)$. *(2 marks)*

$C(n) =$ 3C(n-1) + 1

(b) Write down the boundary condition for the recurrence relation. *(1 marks)*

$C(1) =$ 1

(c) Using the recurrence relation to compute $C(2)$, $C(3)$ and $C(4)$. *(3 marks)*

$C(2) =$ 4

$C(3) =$

$C(4) =$

(d) Prove by induction that $f(n) = \frac{3^n - 1}{2}$ satisfies the recurrence relation for $C(n)$. *(5 marks)*

1 Consider the program (valid for inputs $n \geq 1$)

```
foo(int n) {
   bar();
   if (n==1)
      return;
   foo(n-1);
   foo(n-1);
   foo(n-1);
}
```

(a) Let $C(n)$ be the number of times the function `bar()` is called when we call `foo(n)`. Write down a recurrence relation for $C(n)$. *(2 marks)*

$C(n) =$ 3C(n-1) + 1

(b) Write down the boundary condition for the recurrence relation. *(1 marks)*

$C(1) =$ 1

(c) Using the recurrence relation to compute $C(2)$, $C(3)$ and $C(4)$. *(3 marks)*

$C(2) =$ 4

$C(3) =$ 13

$C(4) =$

(d) Prove by induction that $f(n) = \frac{3^n - 1}{2}$ satisfies the recurrence relation for $C(n)$. *(5 marks)*

1 Consider the program (valid for inputs $n \geq 1$)

```
foo(int n) {
   bar();
   if (n==1)
      return;
   foo(n-1);
   foo(n-1);
   foo(n-1);
}
```

(a) Let $C(n)$ be the number of times the function `bar()` is called when we call `foo(n)`. Write down a recurrence relation for $C(n)$. *(2 marks)*

$C(n) =$ 3C(n-1) + 1

(b) Write down the boundary condition for the recurrence relation. *(1 marks)*

$C(1) =$ 1

(c) Using the recurrence relation to compute $C(2)$, $C(3)$ and $C(4)$. *(3 marks)*

$C(2) =$ 4

$C(3) =$ 13

$C(4) =$ 40

(d) Prove by induction that $f(n) = \frac{3^n - 1}{2}$ satisfies the recurrence relation for $C(n)$. *(5 marks)*

1 Consider the program (valid for inputs $n \geq 1$)

```
foo(int n) {
   bar();
   if (n==1)
      return;
   foo(n-1);
   foo(n-1);
   foo(n-1);
}
```

(a) Let $C(n)$ be the number of times the function `bar()` is called when we call `foo(n)`. Write down a recurrence relation for $C(n)$. *(2 marks)*

$C(n) =$ 3C(n-1) + 1

(b) Write down the boundary condition for the recurrence relation. *(1 marks)*

$C(1) =$ 1

(c) Using the recurrence relation to compute $C(2)$, $C(3)$ and $C(4)$. *(3 marks)*

$C(2) =$ 4

$C(3) =$ 13

$C(4) =$ 40

(d) Prove by induction that $f(n) = \frac{3^n - 1}{2}$ satisfies the recurrence relation for $C(n)$. *(5 marks)*

Question One (d)

Claim: $f(n) = \dfrac{3^n - 1}{2}$ is a solution to $$\begin{gathered} C(n) = 3C(n-1) + 1 \\ C(1) = 1 \end{gathered}$$

Question One (d)

Claim: $f(n) = \dfrac{3^n - 1}{2}$ is a solution to $\begin{array}{c} C(n) = 3C(n-1) + 1 \\ C(1) = 1 \end{array}$

Proof:

We proceed by induction on n to show that C(n) = f(n)

Question One (d)

Claim: $f(n) = \dfrac{3^n - 1}{2}$ is a solution to $\begin{array}{c} C(n) = 3C(n-1) + 1 \\ C(1) = 1 \end{array}$

Proof:

We proceed by induction on n to show that C(n) = f(n)

Base Case: (n=1)

Question One (d)

Claim: $f(n) = \dfrac{3^n - 1}{2}$ is a solution to $$\begin{gathered} C(n) = 3C(n-1) + 1 \\ C(1) = 1 \end{gathered}$$

Proof:

We proceed by induction on n to show that C(n) = f(n)

Base Case: (n=1)

$$C(1) = 1$$

Question One (d)

Claim: $f(n) = \frac{3^n - 1}{2}$ is a solution to $\begin{array}{c} C(n) = 3C(n-1) + 1 \\ C(1) = 1 \end{array}$

Proof:

We proceed by induction on n to show that C(n) = f(n)

Base Case: (n=1)

$$C(1) = 1$$

$$f(1) = \frac{3^1 - 1}{2} = 1$$

Question One (d)

Claim: $f(n) = \dfrac{3^n - 1}{2}$ is a solution to

$$C(n) = 3C(n-1) + 1$$
$$C(1) = 1$$

Proof:

We proceed by induction on n to show that C(n) = f(n)

Base Case: (n=1)

$$C(1) = 1$$

$$f(1) = \frac{3^1 - 1}{2} = 1$$

Therefore $C(1) = f(1)$

as required.

Question One (d)

Claim: $f(n) = \dfrac{3^n - 1}{2}$ is a solution to $$\begin{gathered} C(n) = 3C(n-1) + 1 \\ C(1) = 1 \end{gathered}$$

Proof:

We proceed by induction on n to show that C(n) = f(n)

Base Case: (n=1)

$$C(1) = 1$$

$$f(1) = \frac{3^1 - 1}{2} = 1$$

Therefore $C(1) = f(1)$

as required.

That's the easy bit

Question One (d)

Claim: $f(n) = \dfrac{3^n - 1}{2}$ is a solution to $$\begin{gathered} C(n) = 3C(n-1) + 1 \\ C(1) = 1 \end{gathered}$$

Proof:

We proceed by induction on n to show that C(n) = f(n)

Question One (d)

Claim: $f(n) = \dfrac{3^n - 1}{2}$ is a solution to $$\begin{gathered} C(n) = 3C(n-1) + 1 \\ C(1) = 1 \end{gathered}$$

Proof:

We proceed by induction on n to show that C(n) = f(n)

Step Case: (n>1)

Question One (d)

Claim: $f(n) = \frac{3^n - 1}{2}$ is a solution to $$\begin{gathered} C(n) = 3C(n-1) + 1 \\ C(1) = 1 \end{gathered}$$

Proof:

We proceed by induction on n to show that C(n) = f(n)

Step Case: (n>1)

Suppose that f(n-1) = C(n-1) then

Question One (d)

Claim: $f(n) = \frac{3^n - 1}{2}$ is a solution to $\begin{array}{c} C(n) = 3C(n-1) + 1 \\ C(1) = 1 \end{array}$

Proof:

We proceed by induction on n to show that C(n) = f(n)

Step Case: (n>1)

Suppose that f(n-1) = C(n-1) then

$$C(n) = 3C(n-1) + 1$$ by definition of C

Question One (d)

Claim: $f(n) = \frac{3^n - 1}{2}$ is a solution to $\begin{aligned} C(n) &= 3C(n-1)+1 \\ C(1) &= 1 \end{aligned}$

Proof:

We proceed by induction on n to show that C(n) = f(n)

Step Case: (n>1)

Suppose that f(n-1) = C(n-1) then

$$\begin{aligned} C(n) &= 3C(n-1)+1 && \text{by definition of C} \\ &= 3f(n-1)+1 && \text{by inductive hypothesis} \end{aligned}$$

Question One (d)

Claim: $f(n) = \dfrac{3^n - 1}{2}$ **is a solution to** $$\begin{gathered} C(n) = 3C(n-1) + 1 \\ C(1) = 1 \end{gathered}$$

Proof:

We proceed by induction on n to show that C(n) = f(n)

Step Case: (n>1)

Suppose that f(n-1) = C(n-1) then

$$\begin{aligned} C(n) &= 3C(n-1) + 1 && \text{by definition of C} \\ &= 3f(n-1) + 1 && \text{by inductive hypothesis} \\ &= 3\left(\frac{3^{n-1} - 1}{2}\right) + 1 && \text{by definition of f} \end{aligned}$$

Question One (d)

Claim: $f(n) = \dfrac{3^n - 1}{2}$ is a solution to $\begin{array}{c} C(n) = 3C(n-1) + 1 \\ C(1) = 1 \end{array}$

Proof:

We proceed by induction on n to show that C(n) = f(n)

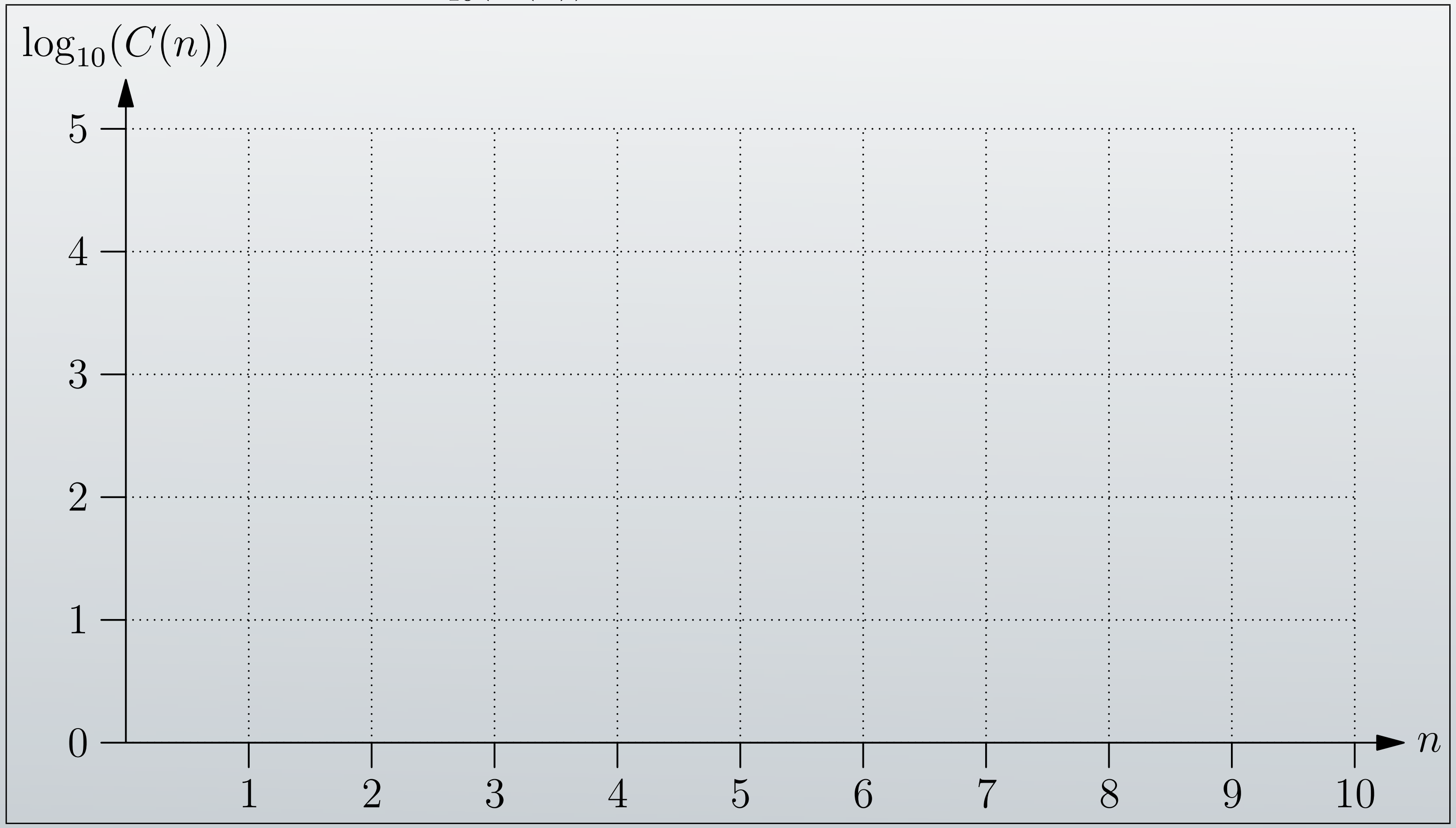
Step Case: (n>1)

Suppose that f(n-1) = C(n-1) then

$$\begin{aligned} C(n) &= 3C(n-1) + 1 && \text{by definition of C} \\ &= 3f(n-1) + 1 && \text{by inductive hypothesis} \\ &= 3\left(\frac{3^{n-1} - 1}{2}\right) + 1 && \text{by definition of f} \\ &= \frac{3^n - 1}{2} = f(n) && \text{by arithmetic and definition of f} \end{aligned}$$

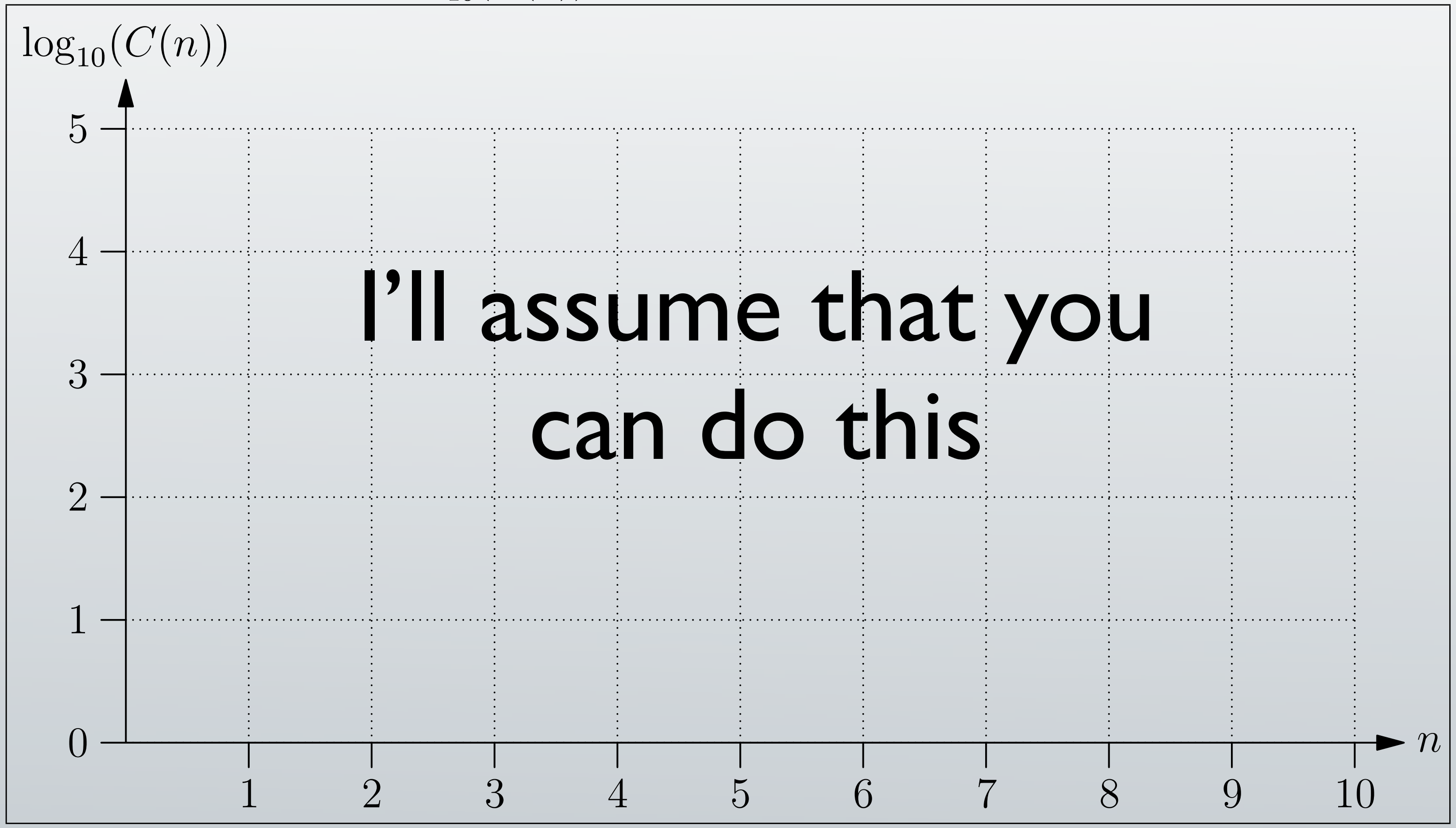
as required.

(e) Sketch the curve $\log_{10}(C(n))$ on the graph below. *(2 marks)*



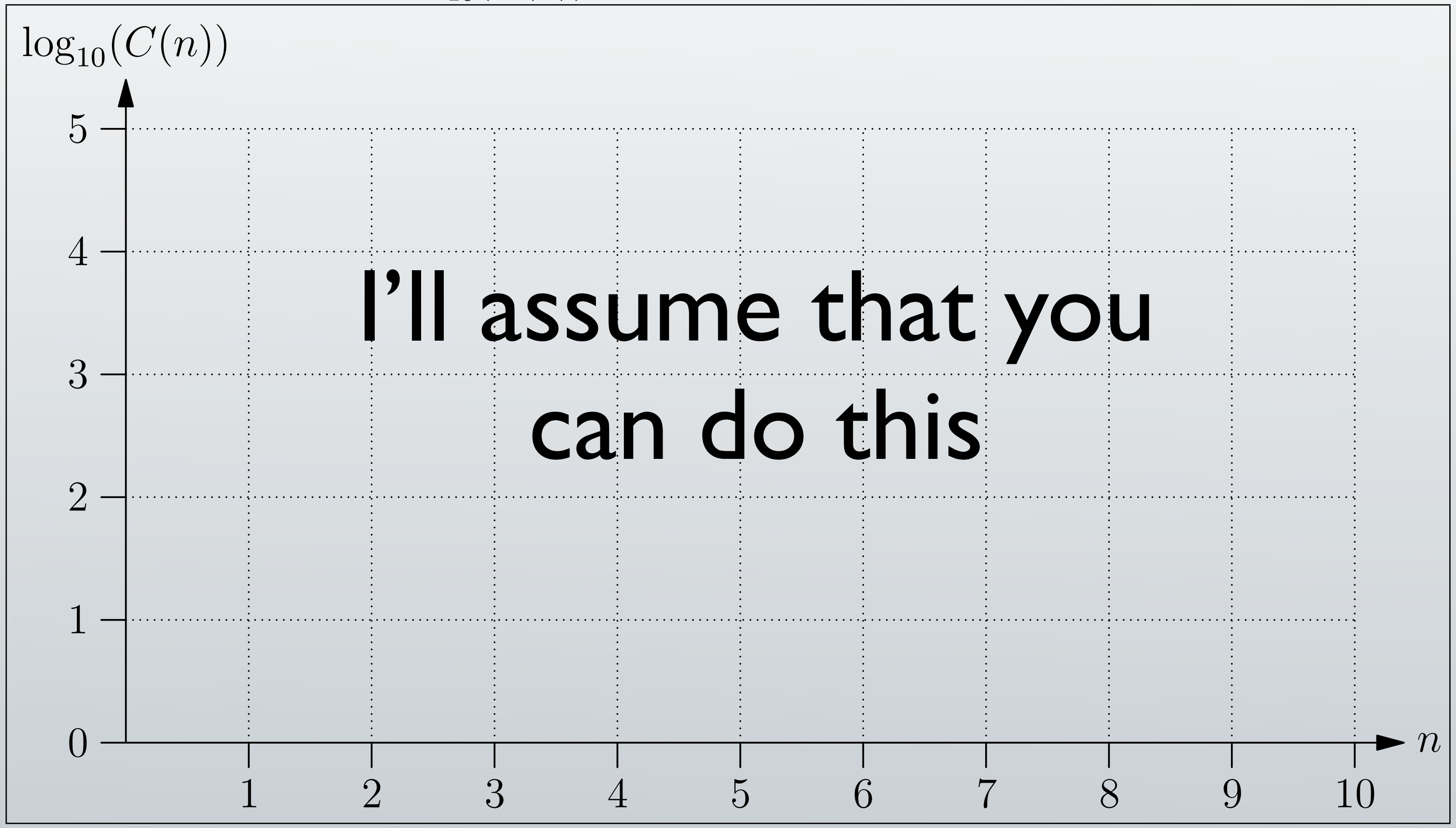
(f) Assume that the most time consuming operation is calling function `bar()` then, if it takes 100s to compute `foo(5)` approximately how long will it take to compute `foo(10)`? *(2 marks)*

(e) Sketch the curve $\log_{10}(C(n))$ on the graph below. *(2 marks)*



(f) Assume that the most time consuming operation is calling function `bar()` then, if it takes 100s to compute `foo(5)` approximately how long will it take to compute `foo(10)`? *(2 marks)*

(e) Sketch the curve $\log_{10}(C(n))$ on the graph below. *(2 marks)*



(f) Assume that the most time consuming operation is calling function `bar()` then, if it takes 100s to compute `foo(5)` approximately how long will it take to compute `foo(10)`? *(2 marks)*

Question One (f)

Time to run foo(n) is approximately $c3^n$ for constant c

Question One (f)

Time to run foo(n) is approximately $c3^n$ for constant c

We know $T(foo(5)) = 100 \approx c \times 3^5$

Question One (f)

Time to run foo(n) is approximately $c3^n$ for constant c

We know $T(foo(5)) = 100 \approx c \times 3^5$

Therefore $c \approx 100 \times 3^{-5}$

Question One (f)

Time to run foo(n) is approximately $c3^n$ for constant c

We know $T(foo(5)) = 100 \approx c \times 3^5$

Therefore $c \approx 100 \times 3^{-5}$

Now $T(foo(10)) \approx c \times 3^{10}$

Question One (f)

Time to run foo(n) is approximately $c3^n$ for constant c

We know $T(foo(5)) = 100 \approx c \times 3^5$

Therefore $c \approx 100 \times 3^{-5}$

Now $T(foo(10)) \approx c \times 3^{10}$

$$\approx (100 \times 3^{-5}) \times 3^{10}$$

Question One (f)

Time to run foo(n) is approximately $c3^n$ for constant c

We know $T(foo(5)) = 100 \approx c \times 3^5$

Therefore $c \approx 100 \times 3^{-5}$

Now $T(foo(10)) \approx c \times 3^{10}$

$$\approx (100 \times 3^{-5}) \times 3^{10}$$

$$= 3^5 \times 100s$$

Question One (f)

Time to run foo(n) is approximately $c3^n$ for constant c

We know $T(foo(5)) = 100 \approx c \times 3^5$

Therefore $c \approx 100 \times 3^{-5}$

Now $T(foo(10)) \approx c \times 3^{10}$

$$\approx (100 \times 3^{-5}) \times 3^{10}$$

$$= 3^5 \times 100s$$

That's about 24,300s

Question Two

2 Consider the program (valid for inputs $n \geq 1$)

```
foo(int n) {
   for(i=1; i<=2n-1; i++)
      bar();
   if (n==1)
      return;
   foo(n-1);
}
```

(a) Let $C(n)$ be the number of times the function `bar()` is called when we call `foo(n)`. Write down a recurrence relation for $C(n)$. *(2 marks)*

$C(n) =$

(b) Write down the boundary condition for the recurrence relation. *(1 marks)*

$C(1) =$

(c) Using the recurrence relation to compute $C(2)$, $C(3)$ and $C(4)$. *(3 marks)*

$C(2) =$

$C(3) =$

$C(4) =$

(d) Guess the solution for $C(n)$ and prove by induction that it satisfies the recurrence relation for $C(n)$. *(5 marks)*

Question Two (a)

How many times is `bar()` called when `foo(n)` is evaluated?

C(n) =

```
foo(int n) {
 for(i=1; i<=2n-1 ;i++)
    { bar(); }
 if (n==1)
    return;
 foo(n-1);
}
```

Question Two (a)

How many times is `bar()` called when `foo(n)` is evaluated?

C(n) =

```
foo(int n) {
 for(i=1; i<=2n-1 ;i++)
     { bar(); }
 if (n==1)
     return;
 foo(n-1);
}
```

Question Two (a)

How many times is `bar()` called when `foo(n)` is evaluated?

once per loop execution

C(n) =

```
foo(int n) {
 for(i=1; i<=2n-1 ;i++)
     { bar(); }
 if (n==1)
   return;
 foo(n-1);
}
```

Question Two (a)

How many times is `bar()` called when `foo(n)` is evaluated?

once per loop execution

$C(n) = \boxed{2n - 1}$

```
foo(int n) {
 for(i=1; i<=2n-1 ;i++)
     { bar(); }
 if (n==1)
    return;
 foo(n-1);
}
```

Question Two (a)

How many times is `bar()` called when `foo(n)` is evaluated?

C(n) = 2n -1

```
foo(int n) {
 for(i=1; i<=2n-1 ;i++)
    { bar(); }
 if (n==1)
    return;
 foo(n-1);
}
```

Question Two (a)

How many times is `bar()` called when `foo(n)` is evaluated?

$C(n) = 2n - 1$

```
foo(int n) {
  for(i=1; i<=2n-1 ;i++)
     { bar(); }
  if (n==1)
     return;
  foo(n-1);
}
```

Question Two (a)

How many times is `bar()` called when `foo(n)` is evaluated?

> How many times is `bar()` called when `foo(n-1)` is evaluated?

$C(n) = 2n - 1$

```
foo(int n) {
 for(i=1; i<=2n-1 ;i++)
    { bar(); }
 if (n==1)
    return;
 foo(n-1);
}
```

Question Two (a)

How many times is `bar()` called when `foo(n)` is evaluated?

How many times is `bar()` called when `foo(n-1)` is evaluated?

$$C(n) = 2n - 1 + \boxed{C(n-1)}$$

```
foo(int n) {
 for(i=1; i<=2n-1 ;i++)
   { bar(); }
 if (n==1)
   return;
 foo(n-1);
}
```

Question Two (a)

How many times is `bar()` called when `foo(n)` is evaluated?

$C(n) = 2n - 1 + C(n-1)$

$= C(n-1) + 2n - 1$

```
foo(int n) {
 for(i=1; i<=2n-1 ;i++)
    { bar(); }
 if (n==1)
    return;
 foo(n-1);
}
```

2 Consider the program (valid for inputs $n \geq 1$)

```
foo(int n) {
   for(i=1; i<=2n-1; i++)
      bar();
   if (n==1)
      return;
   foo(n-1);
}
```

(a) Let $C(n)$ be the number of times the function `bar()` is called when we call `foo(n)`. Write down a recurrence relation for $C(n)$. *(2 marks)*

$C(n) =$ C(n-1) + 2n - 1

(b) Write down the boundary condition for the recurrence relation. *(1 marks)*

$C(1) =$

(c) Using the recurrence relation to compute $C(2)$, $C(3)$ and $C(4)$. *(3 marks)*

$C(2) =$

$C(3) =$

$C(4) =$

(d) Guess the solution for $C(n)$ and prove by induction that it satisfies the recurrence relation for $C(n)$. *(5 marks)*

2 Consider the program (valid for inputs $n \geq 1$)

```
foo(int n) {
   for(i=1; i<=2n-1; i++)
      bar();
   if (n==1)
      return;
   foo(n-1);
}
```

(a) Let $C(n)$ be the number of times the function `bar()` is called when we call `foo(n)`. Write down a recurrence relation for $C(n)$. *(2 marks)*

$C(n) =$ C(n-1) + 2n - 1

(b) Write down the boundary condition for the recurrence relation. *(1 marks)*

$C(1) =$

(c) Using the recurrence relation to compute $C(2)$, $C(3)$ and $C(4)$. *(3 marks)*

$C(2) =$

$C(3) =$

$C(4) =$

(d) Guess the solution for $C(n)$ and prove by induction that it satisfies the recurrence relation for $C(n)$. *(5 marks)*

Question One (b)

How many times is `bar()` called when `foo(1)` is evaluated?

$C(1) = 1$

```
foo(int n) {
 for(i=1; i<=2n-1 ;i++)
    { bar(); }
 if (n==1)
    return;
 foo(n-1);
}
```

2 Consider the program (valid for inputs $n \geq 1$)

```
foo(int n) {
   for(i=1; i<=2n-1; i++)
      bar();
   if (n==1)
      return;
   foo(n-1);
}
```

(a) Let $C(n)$ be the number of times the function `bar()` is called when we call `foo(n)`. Write down a recurrence relation for $C(n)$. *(2 marks)*

$C(n) =$ C(n-1) + 2n - 1

(b) Write down the boundary condition for the recurrence relation. *(1 marks)*

$C(1) =$ 1

(c) Using the recurrence relation to compute $C(2)$, $C(3)$ and $C(4)$. *(3 marks)*

$C(2) =$

$C(3) =$

$C(4) =$

(d) Guess the solution for $C(n)$ and prove by induction that it satisfies the recurrence relation for $C(n)$. *(5 marks)*

2 Consider the program (valid for inputs $n \geq 1$)

```
foo(int n) {
   for(i=1; i<=2n-1; i++)
      bar();
   if (n==1)
      return;
   foo(n-1);
}
```

(a) Let $C(n)$ be the number of times the function `bar()` is called when we call `foo(n)`. Write down a recurrence relation for $C(n)$. *(2 marks)*

$C(n) =$ C(n-1) + 2n - 1

(b) Write down the boundary condition for the recurrence relation. *(1 marks)*

$C(1) =$ 1

(c) Using the recurrence relation to compute $C(2)$, $C(3)$ and $C(4)$. *(3 marks)*

$C(2) =$

$C(3) =$

$C(4) =$

(d) Guess the solution for $C(n)$ and prove by induction that it satisfies the recurrence relation for $C(n)$. *(5 marks)*

2 Consider the program (valid for inputs $n \geq 1$)

```
foo(int n) {
   for(i=1; i<=2n-1; i++)
      bar();
   if (n==1)
      return;
   foo(n-1);
}
```

(a) Let $C(n)$ be the number of times the function `bar()` is called when we call `foo(n)`. Write down a recurrence relation for $C(n)$. *(2 marks)*

$C(n) =$ C(n-1) + 2n - 1

(b) Write down the boundary condition for the recurrence relation. *(1 marks)*

$C(1) =$ 1

(c) Using the recurrence relation to compute $C(2)$, $C(3)$ and $C(4)$. *(3 marks)*

$C(2) =$ 4

$C(3) =$

$C(4) =$

(d) Guess the solution for $C(n)$ and prove by induction that it satisfies the recurrence relation for $C(n)$. *(5 marks)*

2 Consider the program (valid for inputs $n \geq 1$)

```
foo(int n) {
   for(i=1; i<=2n-1; i++)
      bar();
   if (n==1)
      return;
   foo(n-1);
}
```

(a) Let $C(n)$ be the number of times the function `bar()` is called when we call `foo(n)`. Write down a recurrence relation for $C(n)$. *(2 marks)*

$C(n) =$ C(n-1) + 2n - 1

(b) Write down the boundary condition for the recurrence relation. *(1 marks)*

$C(1) =$ 1

(c) Using the recurrence relation to compute $C(2)$, $C(3)$ and $C(4)$. *(3 marks)*

$C(2) =$ 4

$C(3) =$ 9

$C(4) =$

(d) Guess the solution for $C(n)$ and prove by induction that it satisfies the recurrence relation for $C(n)$. *(5 marks)*

2 Consider the program (valid for inputs $n \geq 1$)

```
foo(int n) {
   for(i=1; i<=2n-1; i++)
      bar();
   if (n==1)
      return;
   foo(n-1);
}
```

(a) Let $C(n)$ be the number of times the function `bar()` is called when we call `foo(n)`. Write down a recurrence relation for $C(n)$. *(2 marks)*

$C(n) =$ C(n-1) + 2n - 1

(b) Write down the boundary condition for the recurrence relation. *(1 marks)*

$C(1) =$ 1

(c) Using the recurrence relation to compute $C(2)$, $C(3)$ and $C(4)$. *(3 marks)*

$C(2) =$ 4

$C(3) =$ 9

$C(4) =$ 16

(d) Guess the solution for $C(n)$ and prove by induction that it satisfies the recurrence relation for $C(n)$. *(5 marks)*

2 Consider the program (valid for inputs $n \geq 1$)

```
foo(int n) {
   for(i=1; i<=2n-1; i++)
      bar();
   if (n==1)
      return;
   foo(n-1);
}
```

(a) Let $C(n)$ be the number of times the function `bar()` is called when we call `foo(n)`. Write down a recurrence relation for $C(n)$. *(2 marks)*

$C(n) =$ C(n-1) + 2n - 1

(b) Write down the boundary condition for the recurrence relation. *(1 marks)*

$C(1) =$ 1

(c) Using the recurrence relation to compute $C(2)$, $C(3)$ and $C(4)$. *(3 marks)*

$C(2) =$ 4

$C(3) =$ 9

$C(4) =$ 16

(d) Guess the solution for $C(n)$ and prove by induction that it satisfies the recurrence relation for $C(n)$. *(5 marks)*

2 Consider the program (valid for inputs $n \geq 1$)

```
foo(int n) {
   for(i=1; i<=2n-1; i++)
      bar();
   if (n==1)
      return;
   foo(n-1);
}
```

(a) Let $C(n)$ be the number of times the function `bar()` is called when we call `foo(n)`. Write down a recurrence relation for $C(n)$. *(2 marks)*

$C(n) =$ C(n-1) + 2n - 1

(b) Write down the boundary condition for the recurrence relation. *(1 marks)*

$C(1) =$ 1

(c) Using the recurrence relation to compute $C(2)$, $C(3)$ and $C(4)$. *(3 marks)*

$C(2) =$ 4

$C(3) =$ 9

$C(4) =$ 16

(d) Guess the solution for $C(n)$ and prove by induction that it satisfies the recurrence relation for $C(n)$. *(5 marks)*

I really hope you can guess the solution

Question Two (d)

Claim: $f(n) = n^2$ is a solution to

$$C(n) = C(n-1) + 2n - 1$$
$$C(1) = 1$$

Proof:

We proceed by induction on n to show that C(n) = f(n)

Base Case: (n=1)

$$C(1) = 1$$

$$f(1) = 1^2 = 1$$

Therefore $$C(1) = f(1)$$

as required.

That's the easy bit

Question Two (d)

Claim: $f(n) = n^2$ is a solution to $C(n) = C(n-1) + 2n - 1$, $C(1) = 1$

Proof:

We proceed by induction on n to show that C(n) = f(n)

Step Case: (n>1)

Suppose that f(n-1) = C(n-1) then

$$C(n) = C(n-1) + 2n - 1$$ by definition of C

Question Two (d)

Claim: $f(n) = n^2$ is a solution to $$C(n) = C(n-1) + 2n - 1$$ $$C(1) = 1$$

Proof:

We proceed by induction on n to show that C(n) = f(n)

Step Case: (n>1)

Suppose that f(n-1) = C(n-1) then

$$C(n) = C(n-1) + 2n - 1 \qquad \text{by definition of C}$$

$$= f(n-1) + 2n - 1 \qquad \text{by inductive hypothesis}$$

Question Two (d)

Claim: $f(n) = n^2$ is a solution to $C(n) = C(n-1) + 2n - 1$
$C(1) = 1$

Proof:

We proceed by induction on n to show that C(n) = f(n)

Step Case: (n>1)

Suppose that f(n-1) = C(n-1) then

$$C(n) = C(n-1) + 2n - 1 \quad \text{by definition of C}$$

$$= f(n-1) + 2n - 1 \quad \text{by inductive hypothesis}$$

$$= (n-1)^2 + 2n - 1 \quad \text{by definition of f}$$

Question Two (d)

Claim: $f(n) = n^2$ is a solution to
$$C(n) = C(n-1) + 2n - 1$$
$$C(1) = 1$$

Proof:

We proceed by induction on n to show that C(n) = f(n)

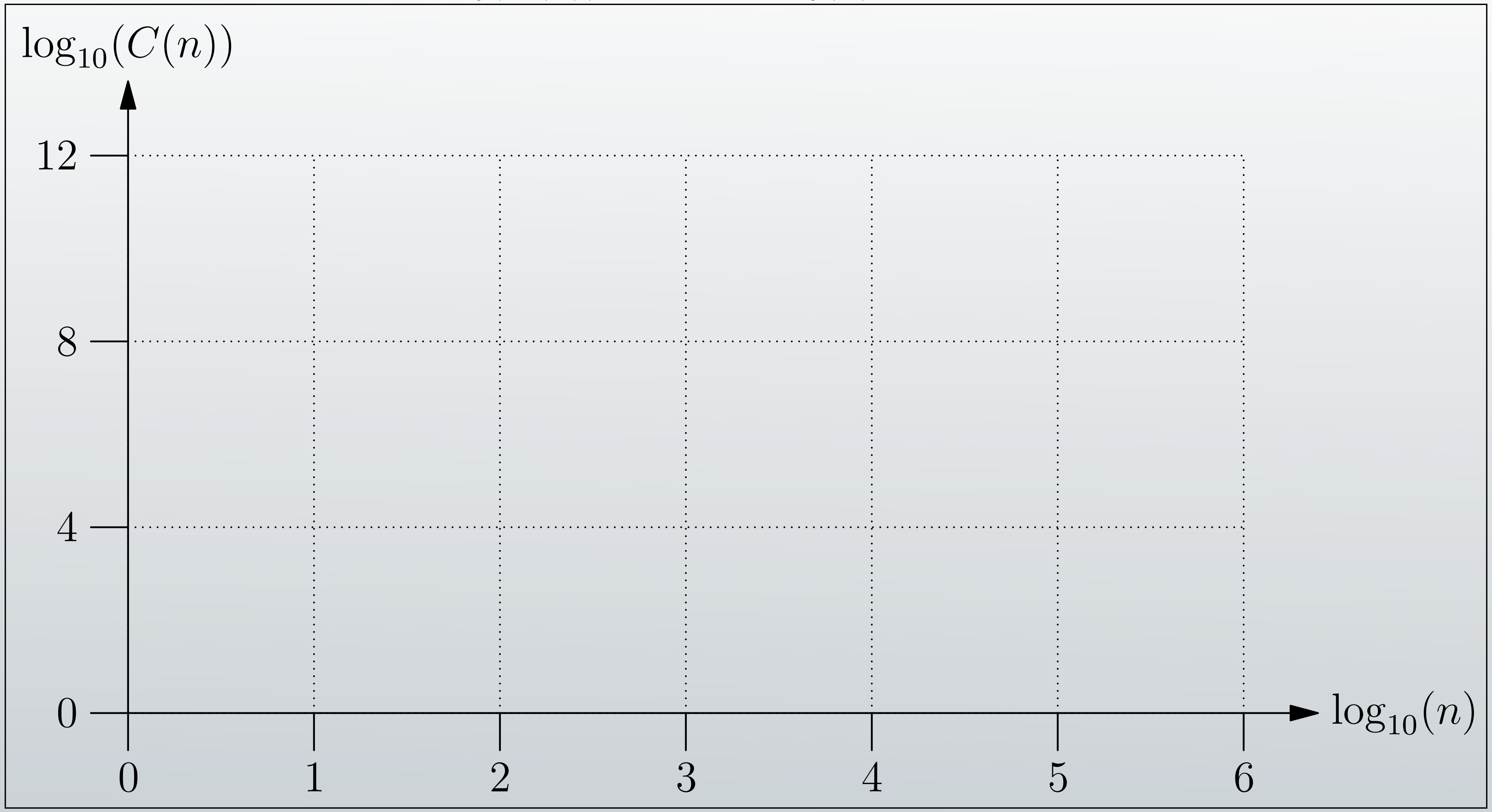
Step Case: (n>1)

Suppose that f(n-1) = C(n-1) then

$C(n) = C(n-1) + 2n - 1$	by definition of C
$= f(n-1) + 2n - 1$	by inductive hypothesis
$= (n-1)^2 + 2n - 1$	by definition of f
$= n^2 = f(n)$	by arithmetic and definition of f

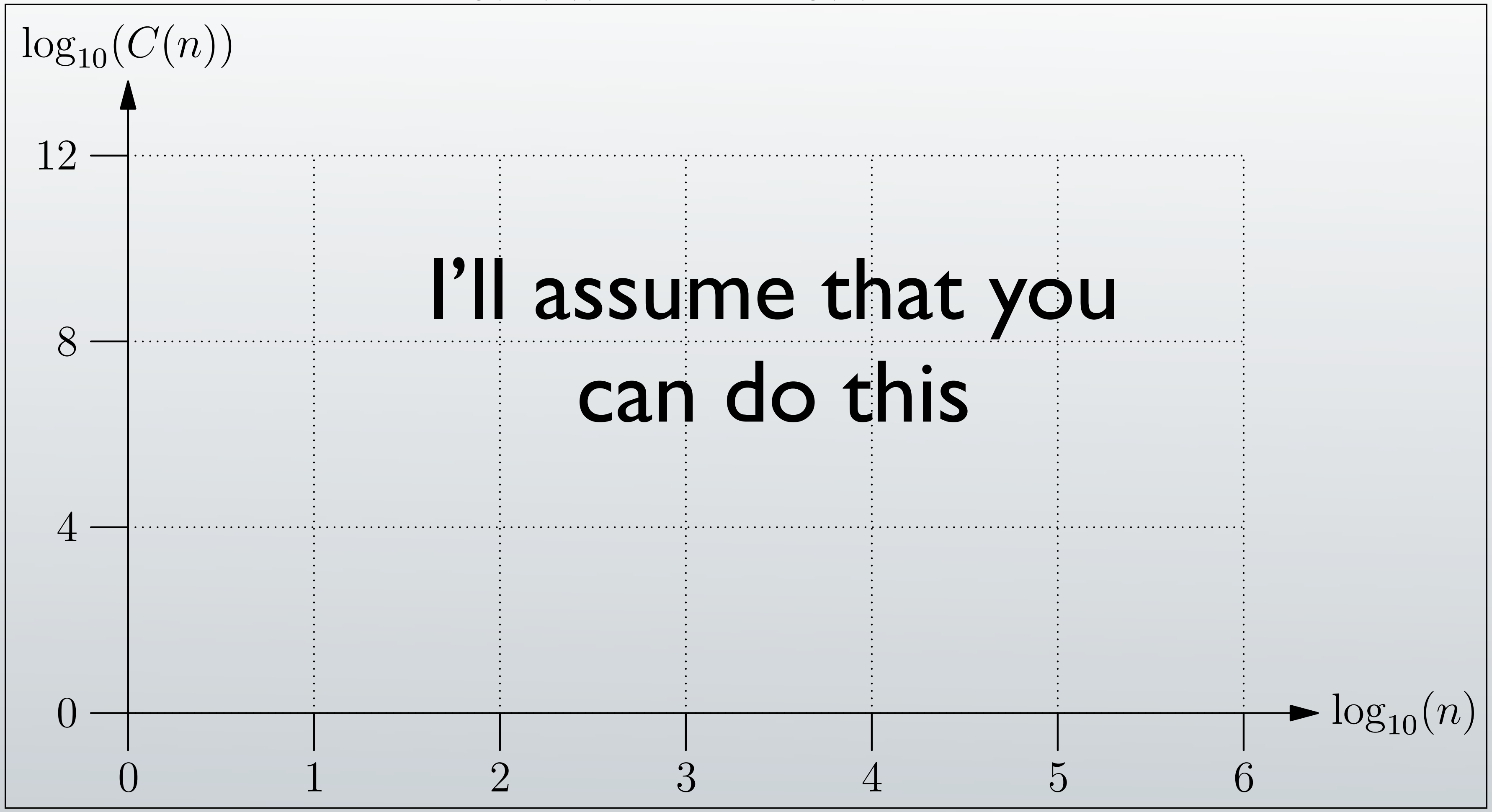
as required.

(e) Sketch the curve $\log_{10}(C(n))$ versus $\log_{10}(n)$ on the graph below. *(2 marks)*



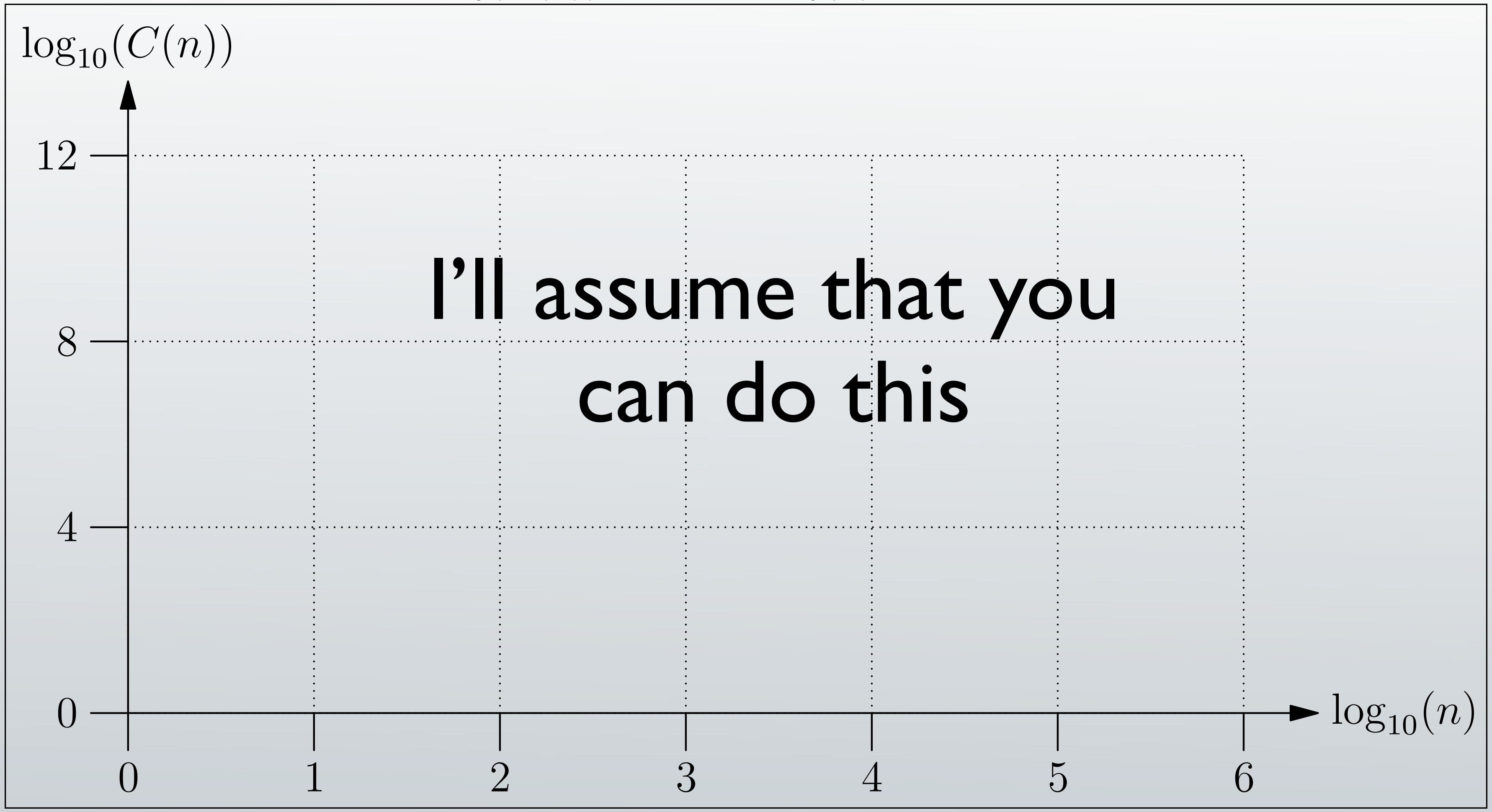
(f) If it takes 100s to compute `foo(1000)` approximately how long will it take to compute `foo(2000)`? *(2 marks)*

(e) Sketch the curve $\log_{10}(C(n))$ versus $\log_{10}(n)$ on the graph below. *(2 marks)*



(f) If it takes 100s to compute `foo(1000)` approximately how long will it take to compute `foo(2000)`? *(2 marks)*

(e) Sketch the curve $\log_{10}(C(n))$ versus $\log_{10}(n)$ on the graph below. *(2 marks)*



I'll assume that you can do this

(f) If it takes 100s to compute `foo(1000)` approximately how long will it take to compute `foo(2000)`? *(2 marks)*

Question Two (f)

Time to run foo(n) is approximately cn^2 for constant c

Question Two (f)

Time to run foo(n) is approximately cn^2 for constant c

We know $T(foo(1000)) = 100 \approx c \times 1000^2$

Question Two (f)

Time to run foo(n) is approximately cn^2 for constant c

We know $T(foo(1000)) = 100 \approx c \times 1000^2$

Therefore $c \approx 100 \times 1000^{-2} = 10^{-4}$

Question Two (f)

Time to run foo(n) is approximately cn^2 for constant c

We know $T(foo(1000)) = 100 \approx c \times 1000^2$

Therefore $c \approx 100 \times 1000^{-2} = 10^{-4}$

Now $T(foo(2000)) \approx c \times 2000^2$

Question Two (f)

Time to run foo(n) is approximately cn^2 for constant c

We know $T(foo(1000)) = 100 \approx c \times 1000^2$

Therefore $c \approx 100 \times 1000^{-2} = 10^{-4}$

Now $T(foo(2000)) \approx c \times 2000^2$

$$\approx (10^{-4}) \times 2000^2$$

Question Two (f)

Time to run foo(n) is approximately cn^2 for constant c

We know $T(foo(1000)) = 100 \approx c \times 1000^2$

Therefore $c \approx 100 \times 1000^{-2} = 10^{-4}$

Now $T(foo(2000)) \approx c \times 2000^2$

$$\approx (10^{-4}) \times 2000^2$$

$$= 400s$$

Question Three

3 Consider the program (valid for inputs $n \geq 1$)

```
foo(int n) {
   bar();
   if (n==1)
      return;
   int m = (int) n/2
   foo(m);
}
```

where `(int) n/2` returns the greatest integer less than or equal to n/2 (i.e. $\lfloor n/2 \rfloor$).

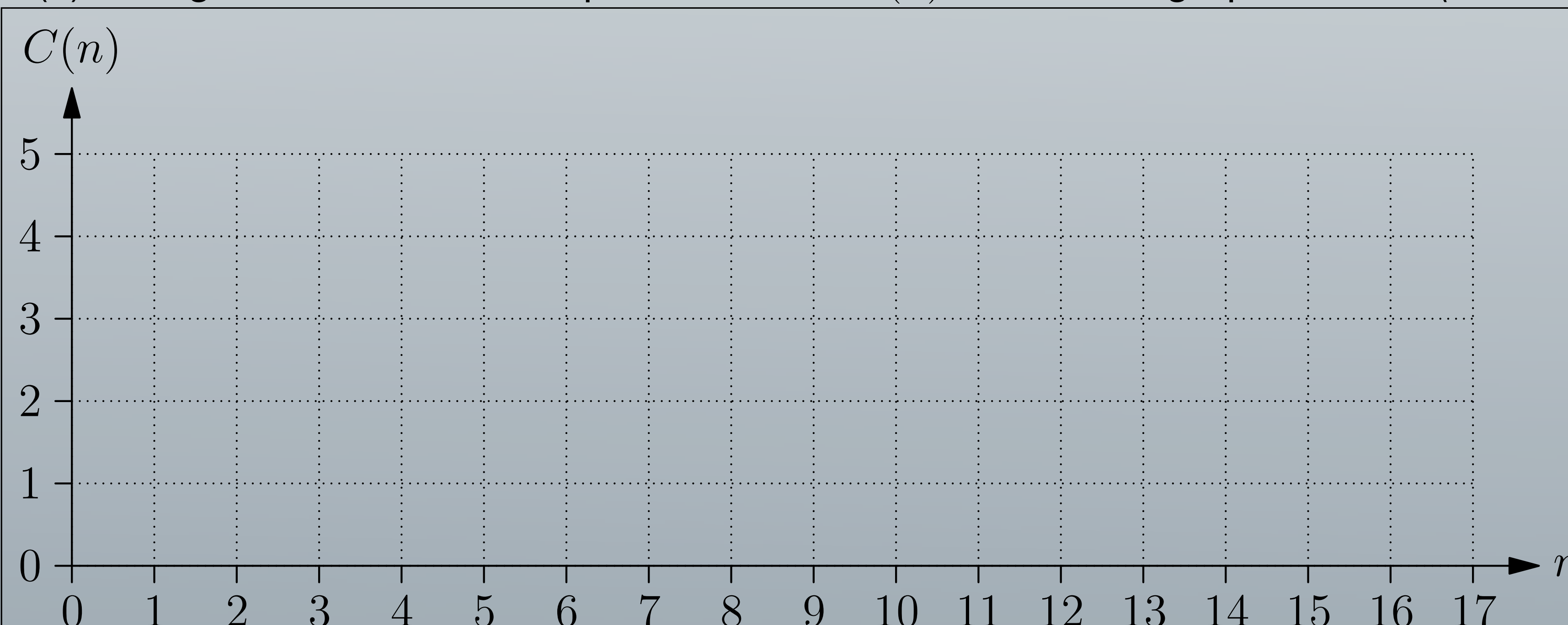
(a) Let $C(n)$ be the number of times the function `bar()` is called when we call `foo(n)`. Write down a recurrence relation for $C(n)$. *(2 marks)*

$$C(n) =$$

(b) Write down the boundary condition for the recurrence relation. *(1 marks)*

$$C(1) =$$

(c) Using the recurrence compute values of $C(n)$ to draw the graph below. *(4 marks)*



3 Consider the program (valid for inputs $n \geq 1$)

```
foo(int n) {
   bar();
   if (n==1)
      return;
   int m = (int) n/2
   foo(m);
}
```

where `(int) n/2` returns the greatest integer less than or equal to n/2 (i.e. $\lfloor n/2 \rfloor$).

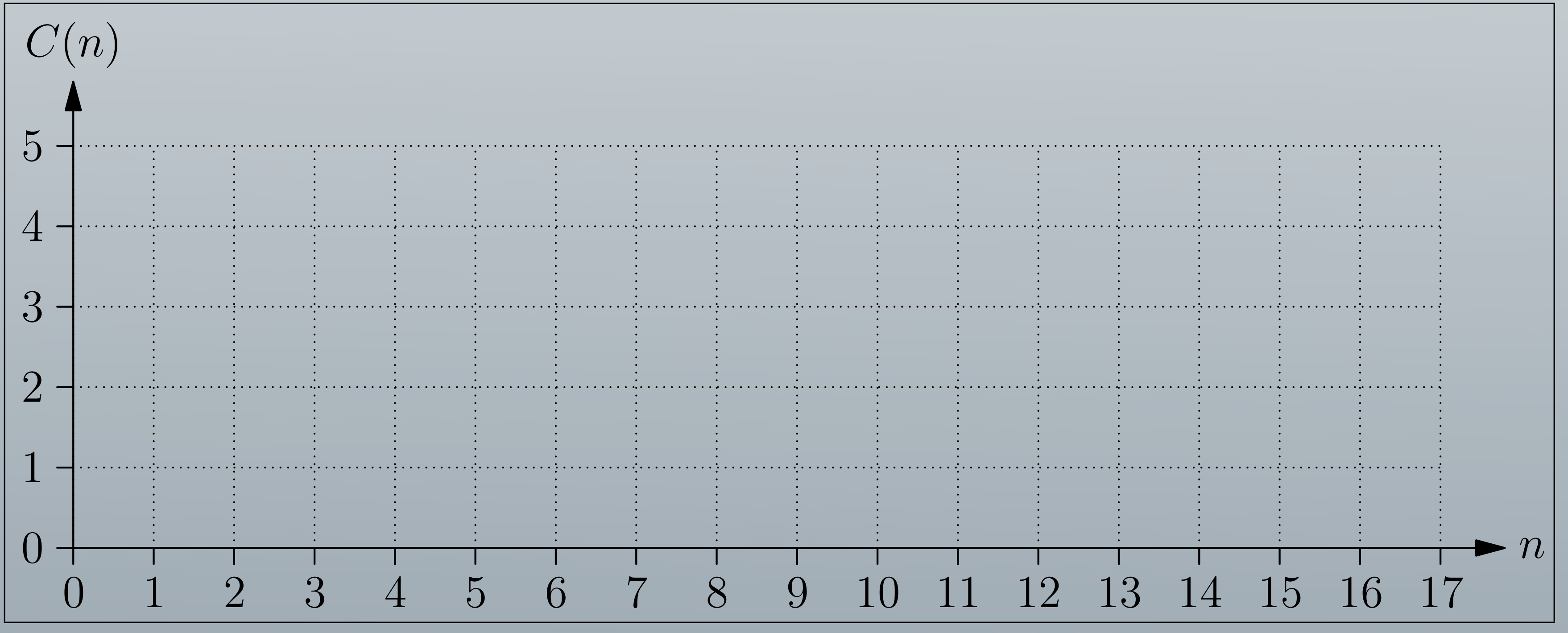
(a) Let $C(n)$ be the number of times the function `bar()` is called when we call `foo(n)`. Write down a recurrence relation for $C(n)$. *(2 marks)*

$C(n) =$

(b) Write down the boundary condition for the recurrence relation. *(1 marks)*

$C(1) =$

(c) Using the recurrence compute values of $C(n)$ to draw the graph below. *(4 marks)*



Question Two (a)

How many times is `bar()` called when `foo(n)` is evaluated?

C(n) =

```
foo(int n) {
  bar();
 if (n==1)
   return;
 int m = (int) n/2;
  foo(m);
}
```

Question Two (a)

How many times is `bar()` called when `foo(n)` is evaluated?

C(n) =

```
foo(int n) {
  bar();
  if (n==1)
    return;
  int m = (int) n/2;
  foo(m);
}
```

Question Two (a)

How many times is `bar()` called when `foo(n)` is evaluated?

$C(n) = 1$

```
foo(int n) {
  bar();
  if (n==1)
    return;
  int m = (int) n/2;
   foo(m);
}
```

Question Two (a)

How many times is `bar()` called when `foo(n)` is evaluated?

$C(n) = 1$

```
foo(int n) {
   bar();
  if (n==1)
     return;
  int m = (int) n/2;
   foo(m);
}
```

Question Two (a)

How many times is `bar()` called when `foo(n)` is evaluated?

C(n) = 1

```
foo(int n) {
   bar();
  if (n==1)
     return;
  int m = (int) n/2;
   foo(m);
}
```

Question Two (a)

How many times is `bar()` called when `foo(n)` is evaluated?

How many times is `bar()` called when `foo(m)` is evaluated?

$C(n) = \quad 1$

```
foo(int n) {
  bar();
 if (n==1)
   return;
 int m = (int) n/2;
 foo(m);
}
```

Question Two (a)

How many times is `bar()` called when `foo(n)` is evaluated?

How many times is `bar()` called when `foo(m)` is evaluated?

$$C(n) = 1 + \boxed{C(\lfloor n/2 \rfloor)}$$

```
foo(int n) {
  bar();
  if (n==1)
    return;
  int m = (int) n/2;
  foo(m);
}
```

Question Two (a)

How many times is `bar()` called when `foo(n)` is evaluated?

$$C(n) = 1 + C(\lfloor n/2 \rfloor)$$

```
foo(int n) {
  bar();
 if (n==1)
   return;
 int m = (int) n/2;
  foo(m);
}
```

3 Consider the program (valid for inputs $n \geq 1$)

```
foo(int n) {
   bar();
   if (n==1)
      return;
   int m = (int) n/2
   foo(m);
}
```

where `(int) n/2` returns the greatest integer less than or equal to n/2 (i.e. $\lfloor n/2 \rfloor$).

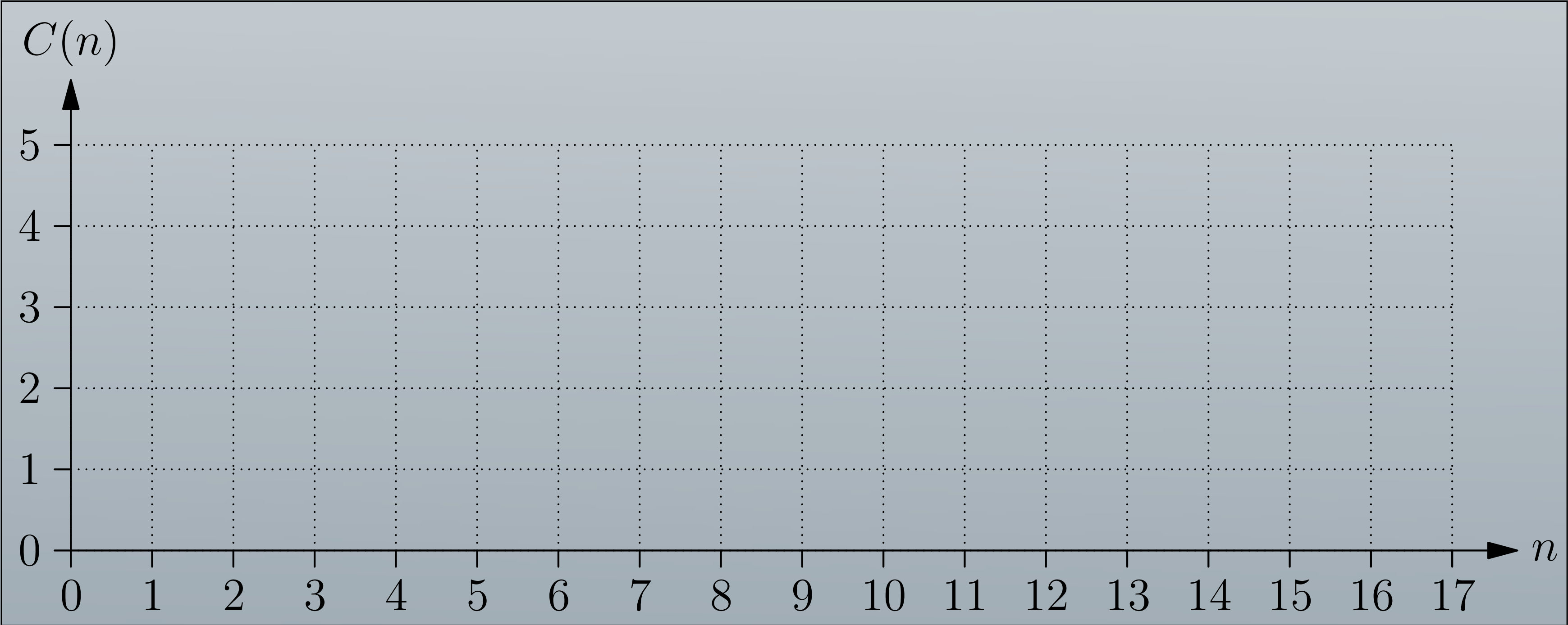
(a) Let $C(n)$ be the number of times the function `bar()` is called when we call `foo(n)`. Write down a recurrence relation for $C(n)$. *(2 marks)*

$C(n) = C(\lfloor n/2 \rfloor) + 1$

(b) Write down the boundary condition for the recurrence relation. *(1 marks)*

$C(1) =$

(c) Using the recurrence compute values of $C(n)$ to draw the graph below. *(4 marks)*



3 Consider the program (valid for inputs $n \geq 1$)

```
foo(int n) {
   bar();
   if (n==1)
      return;
   int m = (int) n/2
   foo(m);
}
```

where `(int) n/2` returns the greatest integer less than or equal to n/2 (i.e. $\lfloor n/2 \rfloor$).

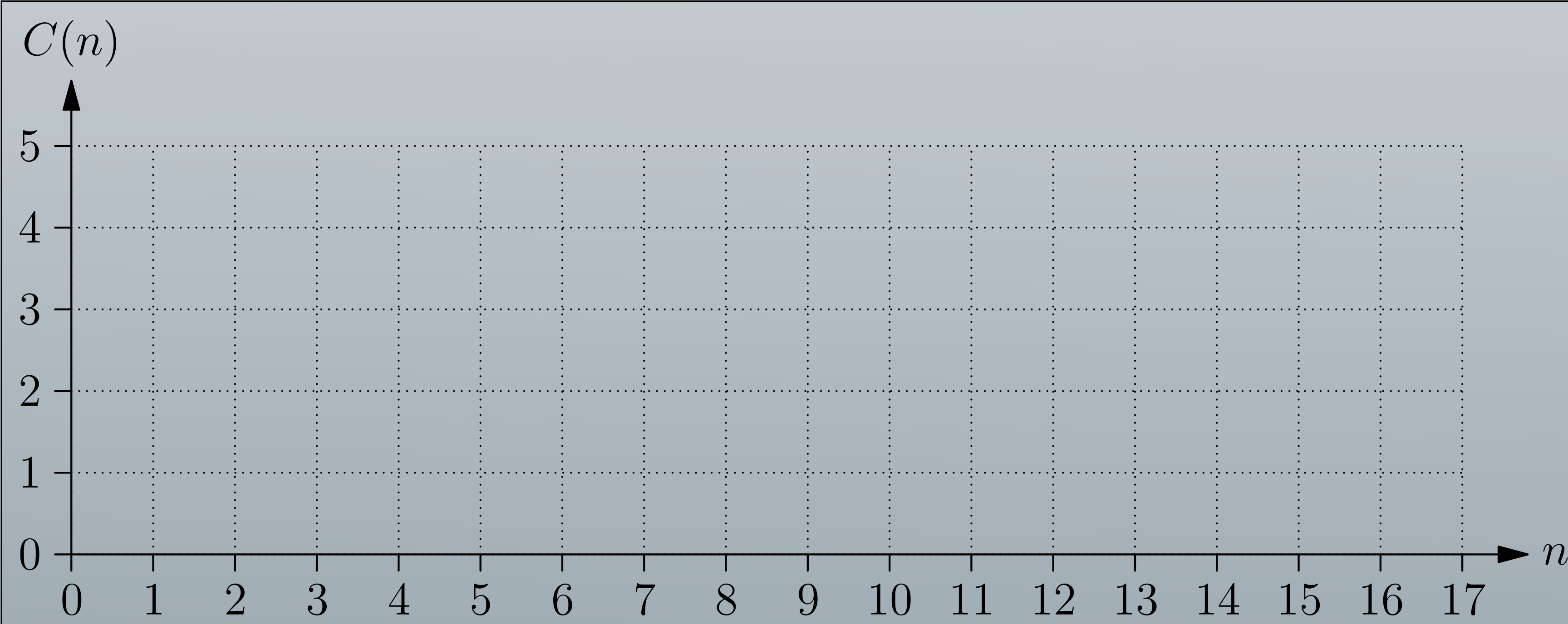
(a) Let $C(n)$ be the number of times the function `bar()` is called when we call `foo(n)`. Write down a recurrence relation for $C(n)$. *(2 marks)*

$C(n) =$ $C(\lfloor n/2 \rfloor) + 1$

(b) Write down the boundary condition for the recurrence relation. *(1 marks)*

$C(1) =$

(c) Using the recurrence compute values of $C(n)$ to draw the graph below. *(4 marks)*



3 Consider the program (valid for inputs $n \geq 1$)

```
foo(int n) {
   bar();
   if (n==1)
      return;
   int m = (int) n/2
   foo(m);
}
```

where `(int) n/2` returns the greatest integer less than or equal to n/2 (i.e. $\lfloor n/2 \rfloor$).

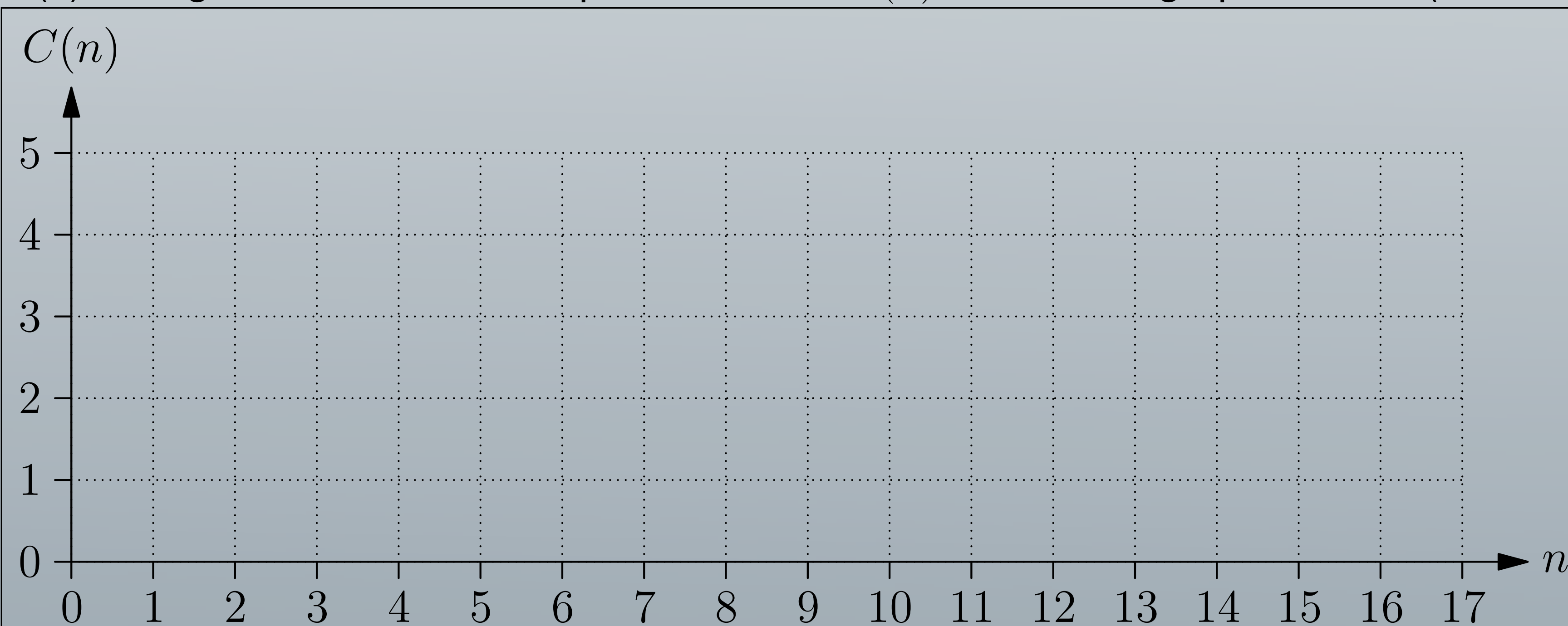
(a) Let $C(n)$ be the number of times the function `bar()` is called when we call `foo(n)`. Write down a recurrence relation for $C(n)$. *(2 marks)*

$C(n) =$ $C(\lfloor n/2 \rfloor) + 1$

(b) Write down the boundary condition for the recurrence relation. *(1 marks)*

$C(1) =$ 1

(c) Using the recurrence compute values of $C(n)$ to draw the graph below. *(4 marks)*



3 Consider the program (valid for inputs $n \geq 1$)

```
foo(int n) {
   bar();
   if (n==1)
      return;
   int m = (int) n/2
   foo(m);
}
```

where `(int) n/2` returns the greatest integer less than or equal to n/2 (i.e. $\lfloor n/2 \rfloor$).

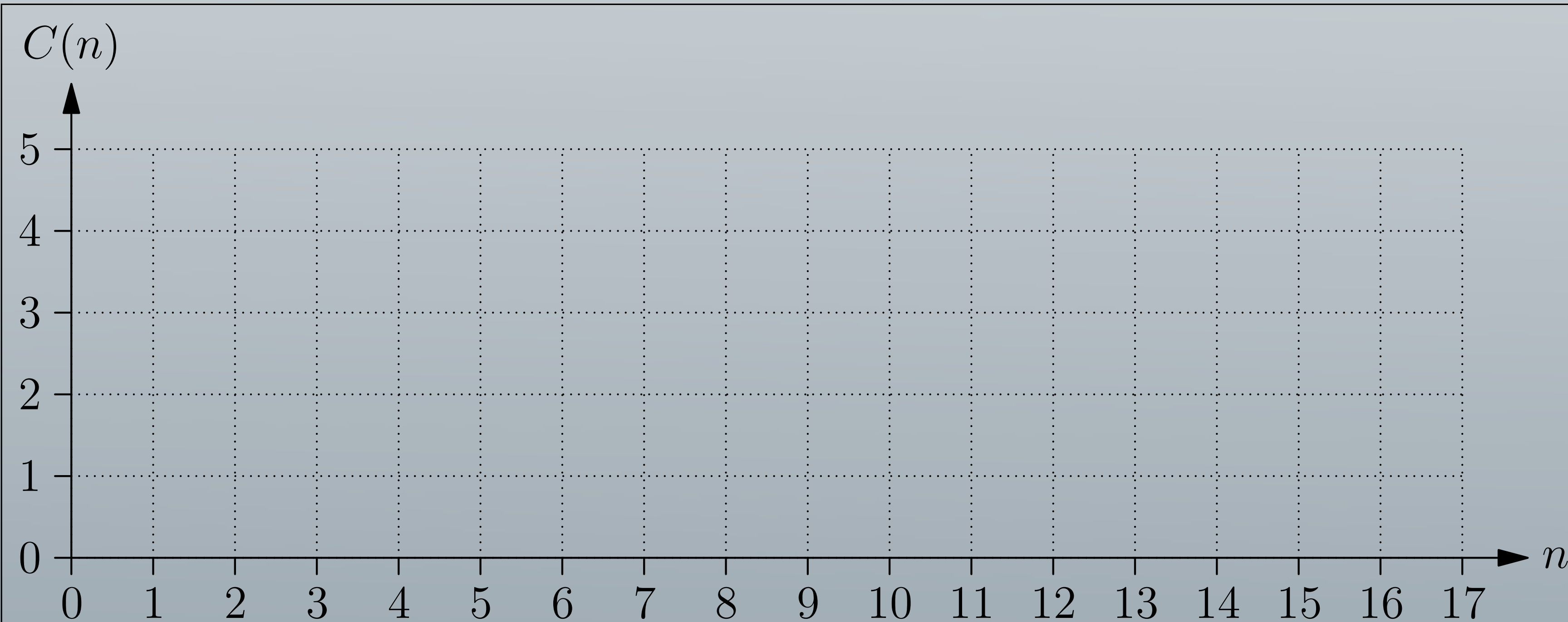
(a) Let $C(n)$ be the number of times the function `bar()` is called when we call `foo(n)`. Write down a recurrence relation for $C(n)$. *(2 marks)*

$C(n) =$ $C(\lfloor n/2 \rfloor) + 1$

(b) Write down the boundary condition for the recurrence relation. *(1 marks)*

$C(1) =$ 1

(c) Using the recurrence compute values of $C(n)$ to draw the graph below. *(4 marks)*



3 Consider the program (valid for inputs $n \geq 1$)

```
foo(int n) {
   bar();
   if (n==1)
      return;
   int m = (int) n/2
   foo(m);
}
```

where `(int) n/2` returns the greatest integer less than or equal to n/2 (i.e. $\lfloor n/2 \rfloor$).

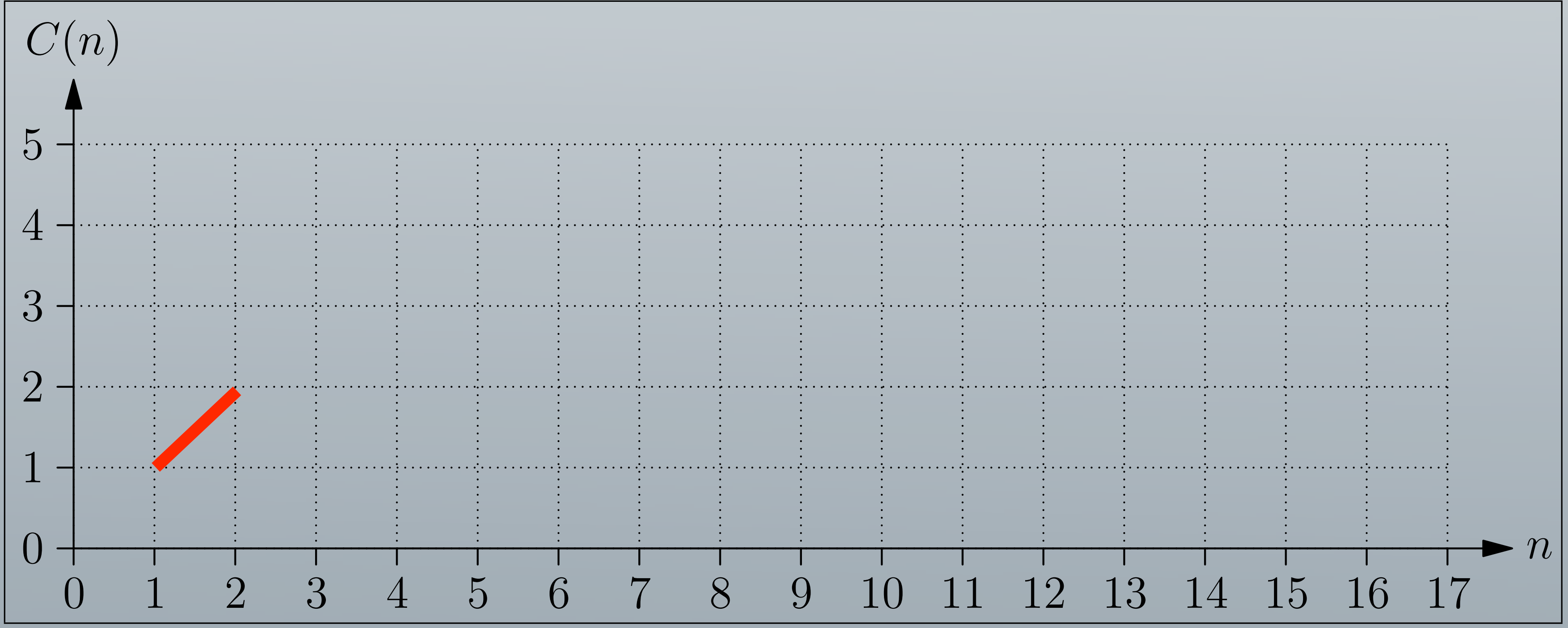
(a) Let $C(n)$ be the number of times the function `bar()` is called when we call `foo(n)`. Write down a recurrence relation for $C(n)$. *(2 marks)*

$C(n) =$ C(⌊n/2⌋) + 1

(b) Write down the boundary condition for the recurrence relation. *(1 marks)*

$C(1) =$ 1

(c) Using the recurrence compute values of $C(n)$ to draw the graph below. *(4 marks)*



3 Consider the program (valid for inputs $n \geq 1$)

```
foo(int n) {
   bar();
   if (n==1)
      return;
   int m = (int) n/2
   foo(m);
}
```

where `(int) n/2` returns the greatest integer less than or equal to n/2 (i.e. $\lfloor n/2 \rfloor$).

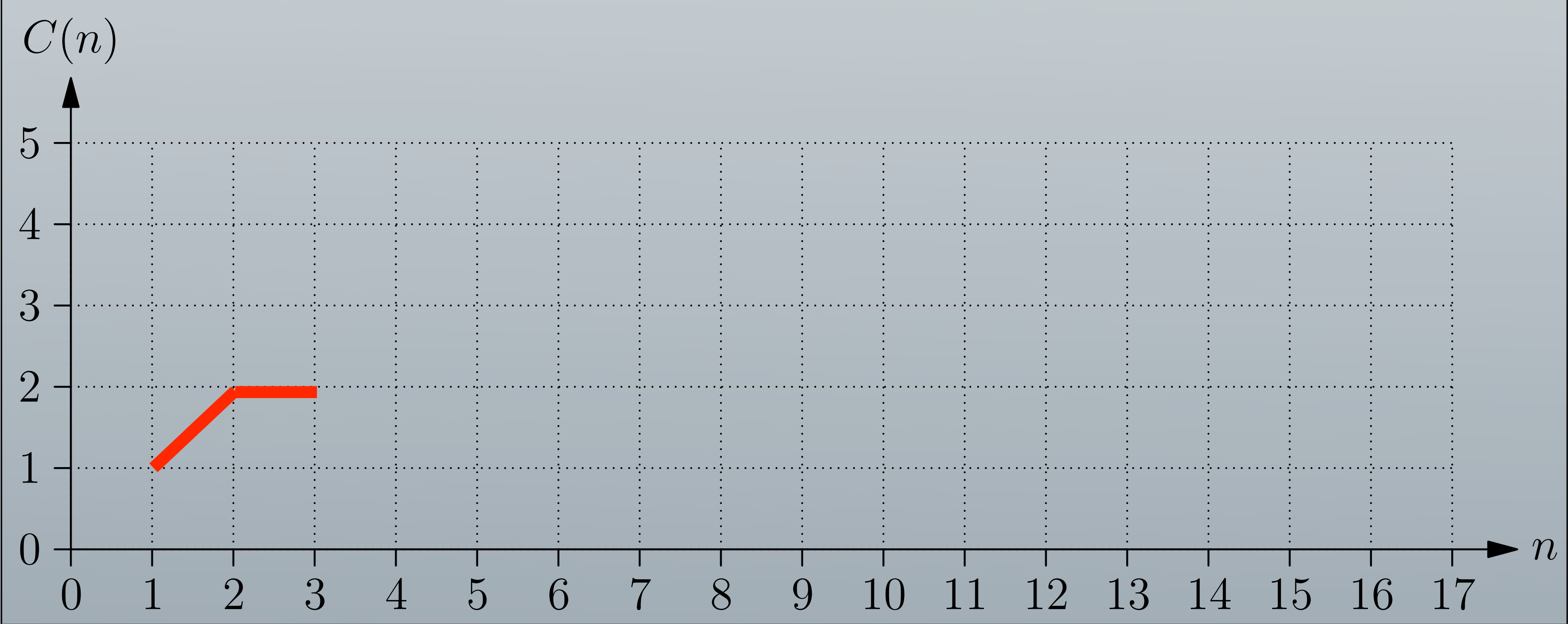
(a) Let $C(n)$ be the number of times the function `bar()` is called when we call `foo(n)`. Write down a recurrence relation for $C(n)$. *(2 marks)*

$C(n) =$ $C(\lfloor n/2 \rfloor) + 1$

(b) Write down the boundary condition for the recurrence relation. *(1 marks)*

$C(1) =$ 1

(c) Using the recurrence compute values of $C(n)$ to draw the graph below. *(4 marks)*



3 Consider the program (valid for inputs $n \geq 1$)

```
foo(int n) {
   bar();
   if (n==1)
      return;
   int m = (int) n/2
   foo(m);
}
```

where `(int) n/2` returns the greatest integer less than or equal to n/2 (i.e. $\lfloor n/2 \rfloor$).

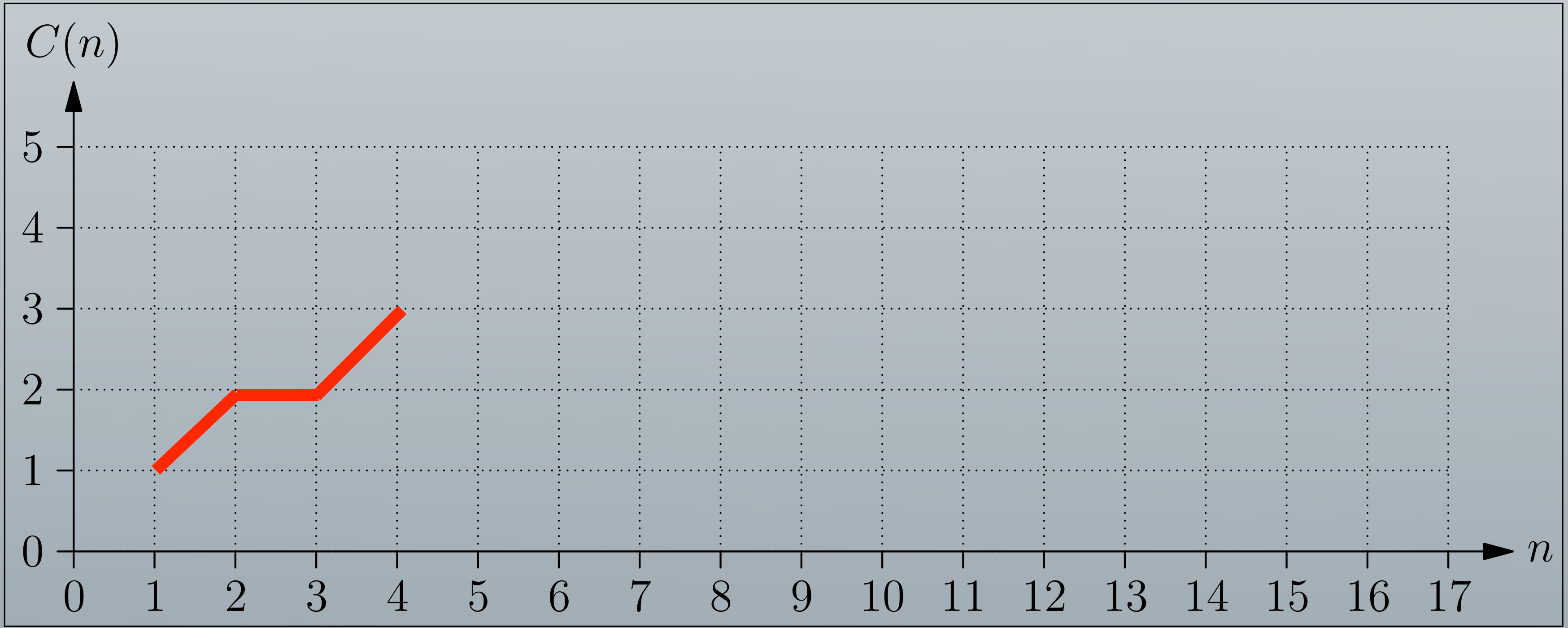
(a) Let $C(n)$ be the number of times the function `bar()` is called when we call `foo(n)`. Write down a recurrence relation for $C(n)$. *(2 marks)*

$C(n) =$ $C(\lfloor n/2 \rfloor) + 1$

(b) Write down the boundary condition for the recurrence relation. *(1 marks)*

$C(1) =$ 1

(c) Using the recurrence compute values of $C(n)$ to draw the graph below. *(4 marks)*



3 Consider the program (valid for inputs $n \geq 1$)

```
foo(int n) {
   bar();
   if (n==1)
      return;
   int m = (int) n/2
   foo(m);
}
```

where `(int) n/2` returns the greatest integer less than or equal to n/2 (i.e. $\lfloor n/2 \rfloor$).

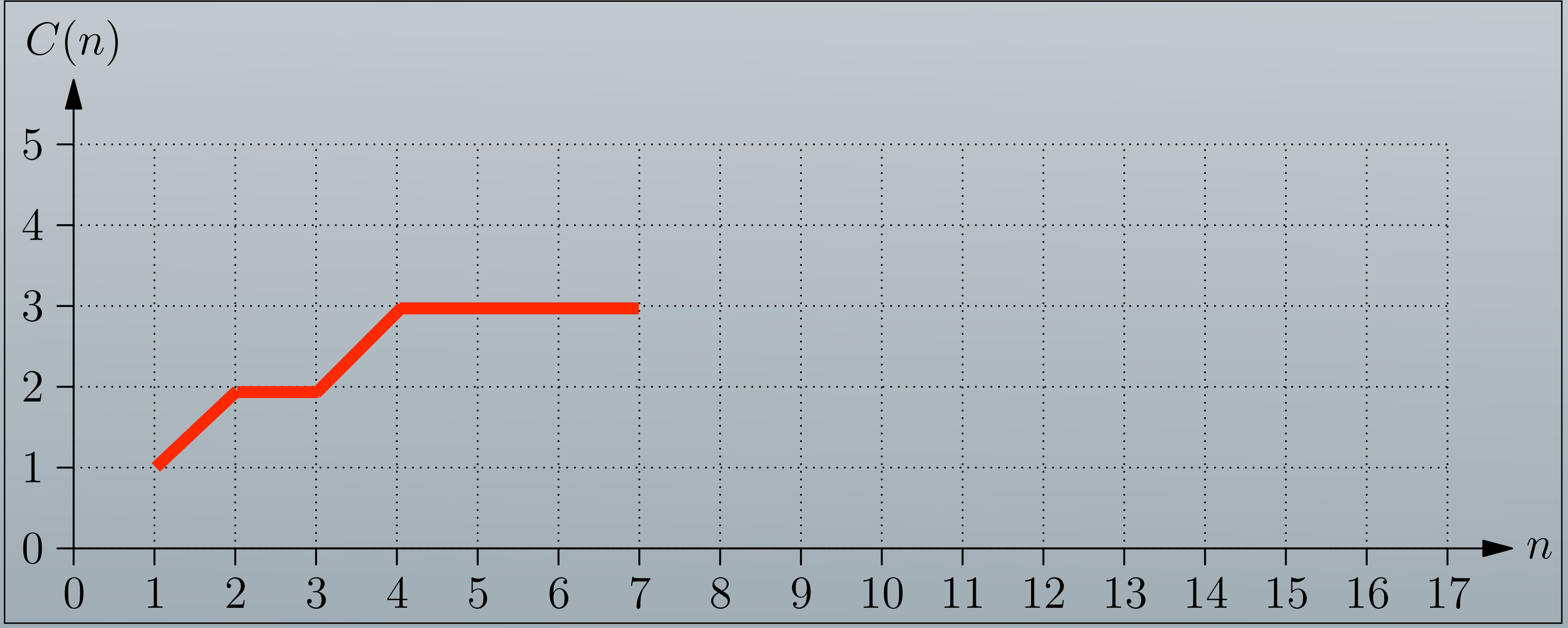
(a) Let $C(n)$ be the number of times the function `bar()` is called when we call `foo(n)`. Write down a recurrence relation for $C(n)$. *(2 marks)*

$C(n) =$ $C(\lfloor n/2 \rfloor) + 1$

(b) Write down the boundary condition for the recurrence relation. *(1 marks)*

$C(1) =$ 1

(c) Using the recurrence compute values of $C(n)$ to draw the graph below. *(4 marks)*



3 Consider the program (valid for inputs $n \geq 1$)

```
foo(int n) {
   bar();
   if (n==1)
      return;
   int m = (int) n/2
   foo(m);
}
```

where `(int) n/2` returns the greatest integer less than or equal to n/2 (i.e. $\lfloor n/2 \rfloor$).

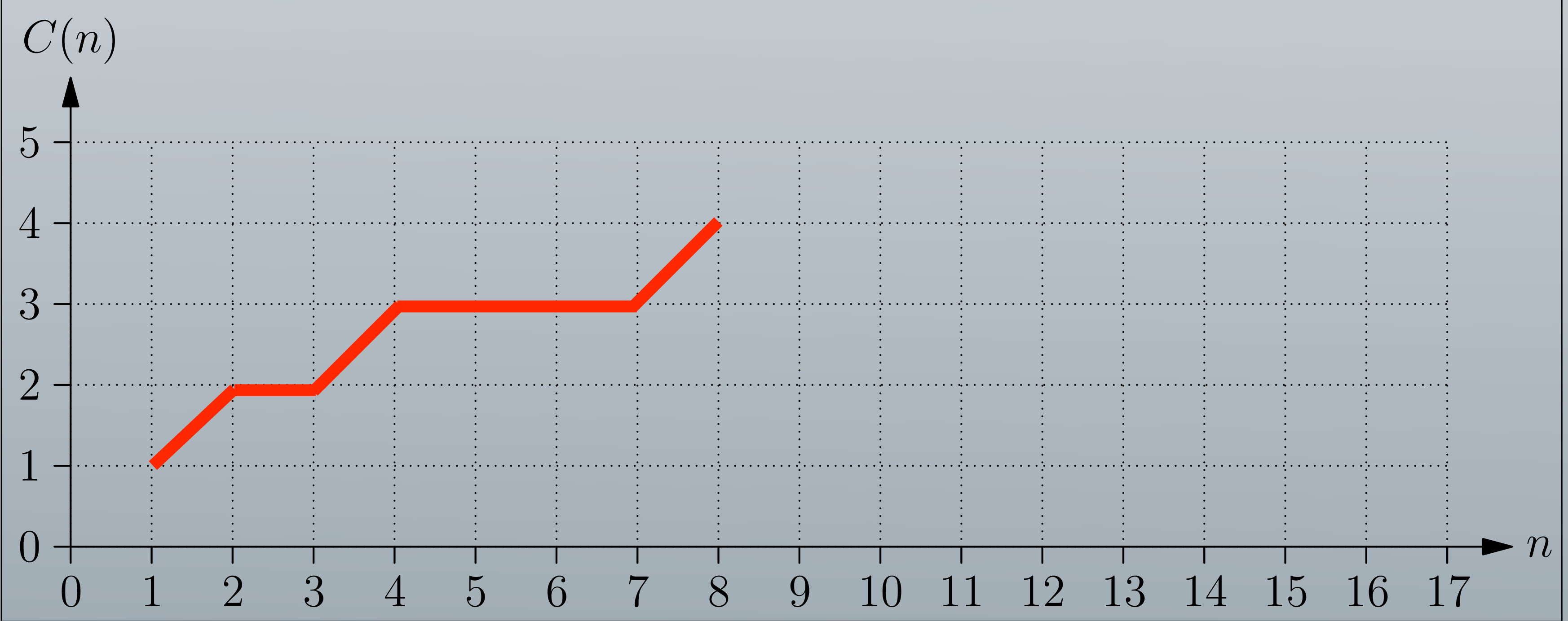
(a) Let $C(n)$ be the number of times the function `bar()` is called when we call `foo(n)`. Write down a recurrence relation for $C(n)$. *(2 marks)*

$C(n) =$ C(⌊n/2⌋) + 1

(b) Write down the boundary condition for the recurrence relation. *(1 marks)*

$C(1) =$ 1

(c) Using the recurrence compute values of $C(n)$ to draw the graph below. *(4 marks)*



3 Consider the program (valid for inputs $n \geq 1$)

```
foo(int n) {
   bar();
   if (n==1)
      return;
   int m = (int) n/2
   foo(m);
}
```

where `(int) n/2` returns the greatest integer less than or equal to n/2 (i.e. $\lfloor n/2 \rfloor$).

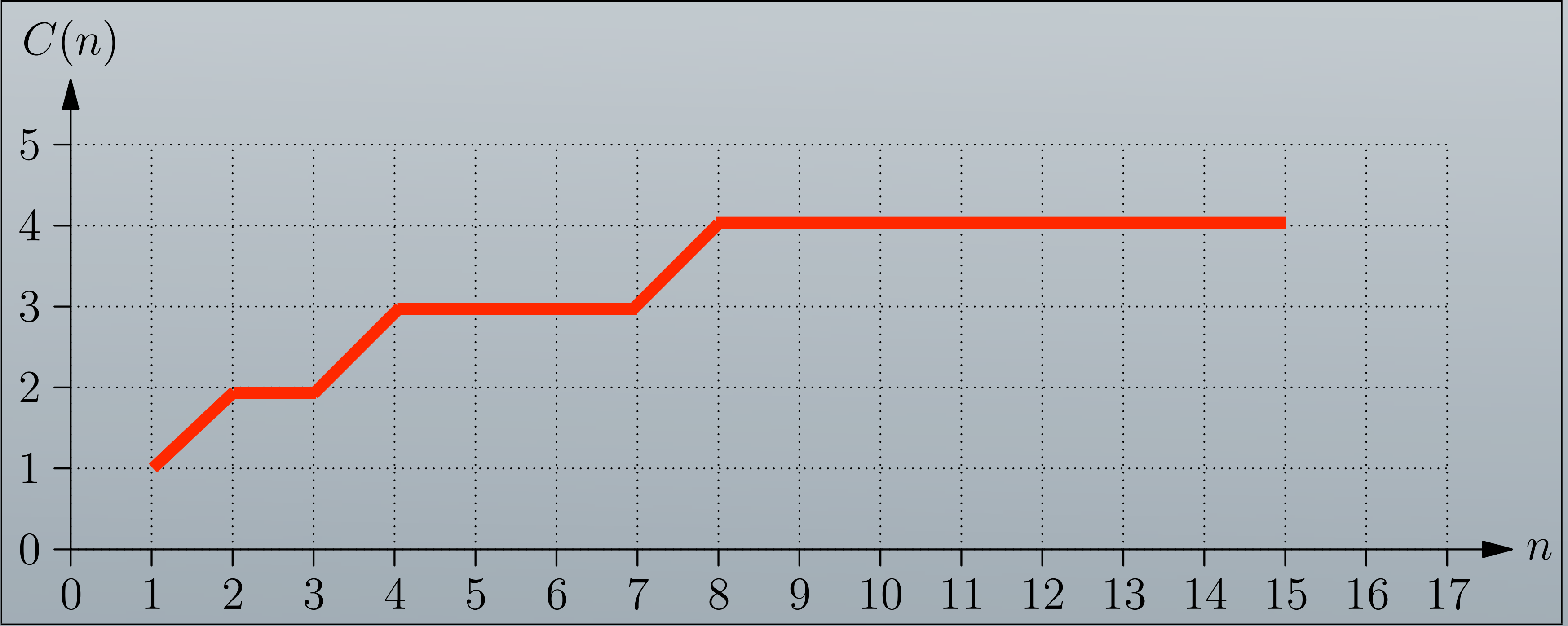
(a) Let $C(n)$ be the number of times the function `bar()` is called when we call `foo(n)`. Write down a recurrence relation for $C(n)$. *(2 marks)*

$C(n) =$ C(⌊n/2⌋) + 1

(b) Write down the boundary condition for the recurrence relation. *(1 marks)*

$C(1) =$ 1

(c) Using the recurrence compute values of $C(n)$ to draw the graph below. *(4 marks)*



3 Consider the program (valid for inputs $n \geq 1$)

```
foo(int n) {
   bar();
   if (n==1)
      return;
   int m = (int) n/2
   foo(m);
}
```

where `(int) n/2` returns the greatest integer less than or equal to n/2 (i.e. $\lfloor n/2 \rfloor$).

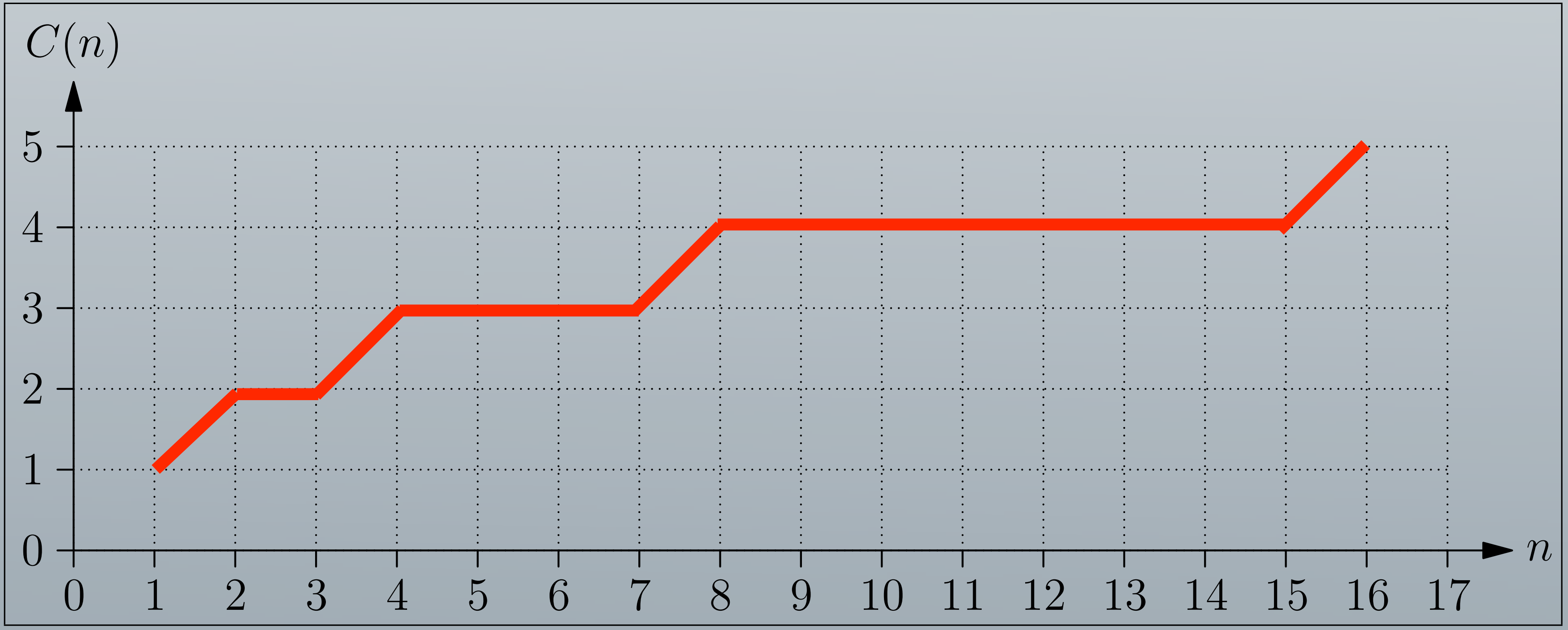
(a) Let $C(n)$ be the number of times the function `bar()` is called when we call `foo(n)`. Write down a recurrence relation for $C(n)$. *(2 marks)*

$C(n) =$ C(⌊n/2⌋) + 1

(b) Write down the boundary condition for the recurrence relation. *(1 marks)*

$C(1) =$ 1

(c) Using the recurrence compute values of $C(n)$ to draw the graph below. *(4 marks)*



3 Consider the program (valid for inputs $n \geq 1$)

```
foo(int n) {
   bar();
   if (n==1)
      return;
   int m = (int) n/2
   foo(m);
}
```

where `(int) n/2` returns the greatest integer less than or equal to n/2 (i.e. $\lfloor n/2 \rfloor$).

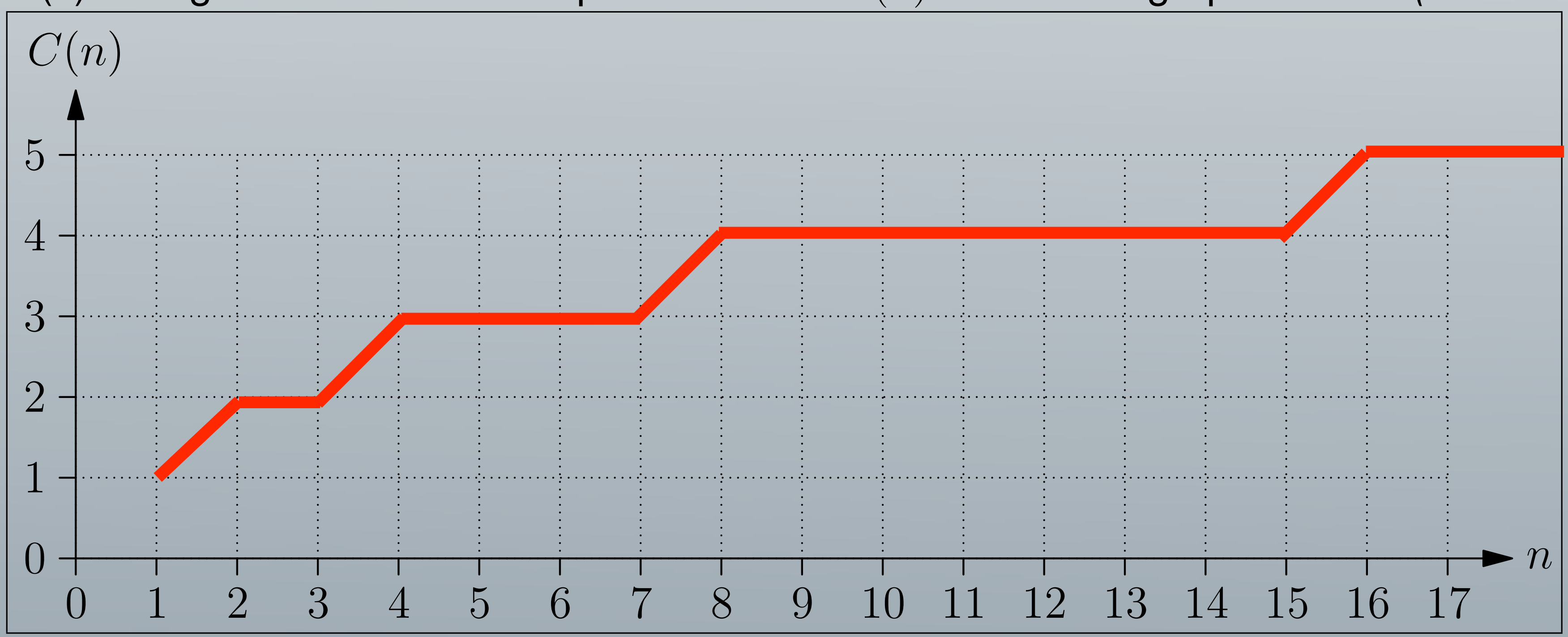
(a) Let $C(n)$ be the number of times the function `bar()` is called when we call `foo(n)`. Write down a recurrence relation for $C(n)$. *(2 marks)*

$C(n) =$ $C(\lfloor n/2 \rfloor) + 1$

(b) Write down the boundary condition for the recurrence relation. *(1 marks)*

$C(1) =$ 1

(c) Using the recurrence compute values of $C(n)$ to draw the graph below. *(4 marks)*



(d) Prove by induction that $f(n) = \lfloor \log_2(n) \rfloor + 1$ satisfies the recurrence relation for $C(n)$. This is simplified if we perform the inductive step over the set of integers $S_m = \{2^m, 2^m + 1, \ldots, 2^{m+1} - 1\}$. *(6 marks)*

(e) If it takes 100s to compute `foo(512)` approximately how long will it take to compute `foo(1024)`? *(2 marks)*

(d) Prove by induction that $f(n) = \lfloor \log_2(n) \rfloor + 1$ satisfies the recurrence relation for $C(n)$. This is simplified if we perform the inductive step over the set of integers $S_m = \{2^m, 2^m + 1, \ldots, 2^{m+1} - 1\}$. *(6 marks)*

(e) If it takes 100s to compute `foo(512)` approximately how long will it take to compute `foo(1024)`? *(2 marks)*

(d) Prove by induction that $f(n) = \lfloor \log_2(n) \rfloor + 1$ satisfies the recurrence relation for $C(n)$. This is simplified if we perform the inductive step over the set of integers $S_m = \{2^m, 2^m + 1, \ldots, 2^{m+1} - 1\}$. *(6 marks)*

What are these S_m sets all about?

(e) If it takes 100s to compute `foo(512)` approximately how long will it take to compute `foo(1024)`? *(2 marks)*

3 Consider the program (valid for inputs $n \geq 1$)

```
foo(int n) {
   bar();
   if (n==1)
      return;
   int m = (int) n/2
   foo(m);
}
```

where `(int) n/2` returns the greatest integer less than or equal to n/2 (i.e. $\lfloor n/2 \rfloor$).

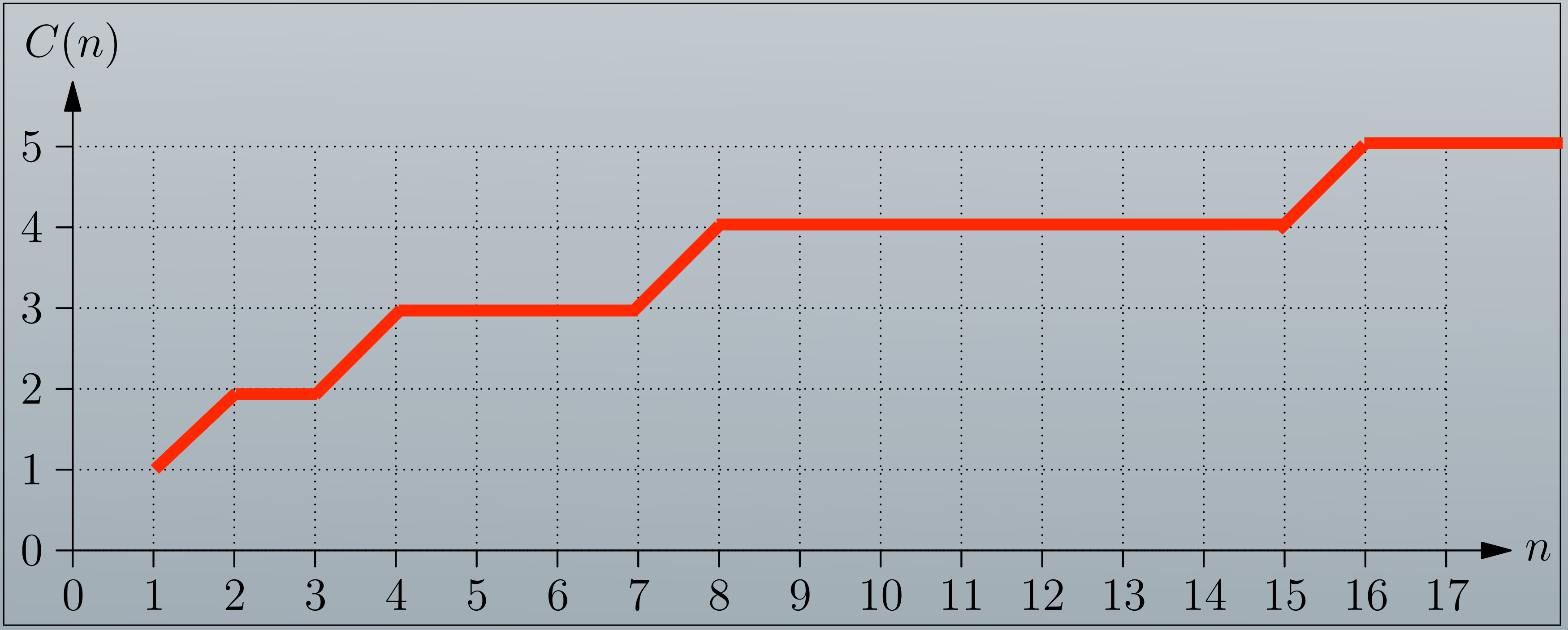
(a) Let $C(n)$ be the number of times the function `bar()` is called when we call `foo(n)`. Write down a recurrence relation for $C(n)$. *(2 marks)*

$C(n) =$ C(⌊n/2⌋) + 1

(b) Write down the boundary condition for the recurrence relation. *(1 marks)*

$C(1) =$ 1

(c) Using the recurrence compute values of $C(n)$ to draw the graph below. *(4 marks)*



3 Consider the program (valid for inputs $n \geq 1$)

$$S_m = \{2^m, 2^m + 1, \ldots, 2^{m+1} - 1\}$$

}

where `(int) n/2` returns the greatest integer less than or equal to n/2 (i.e. $\lfloor n/2 \rfloor$).

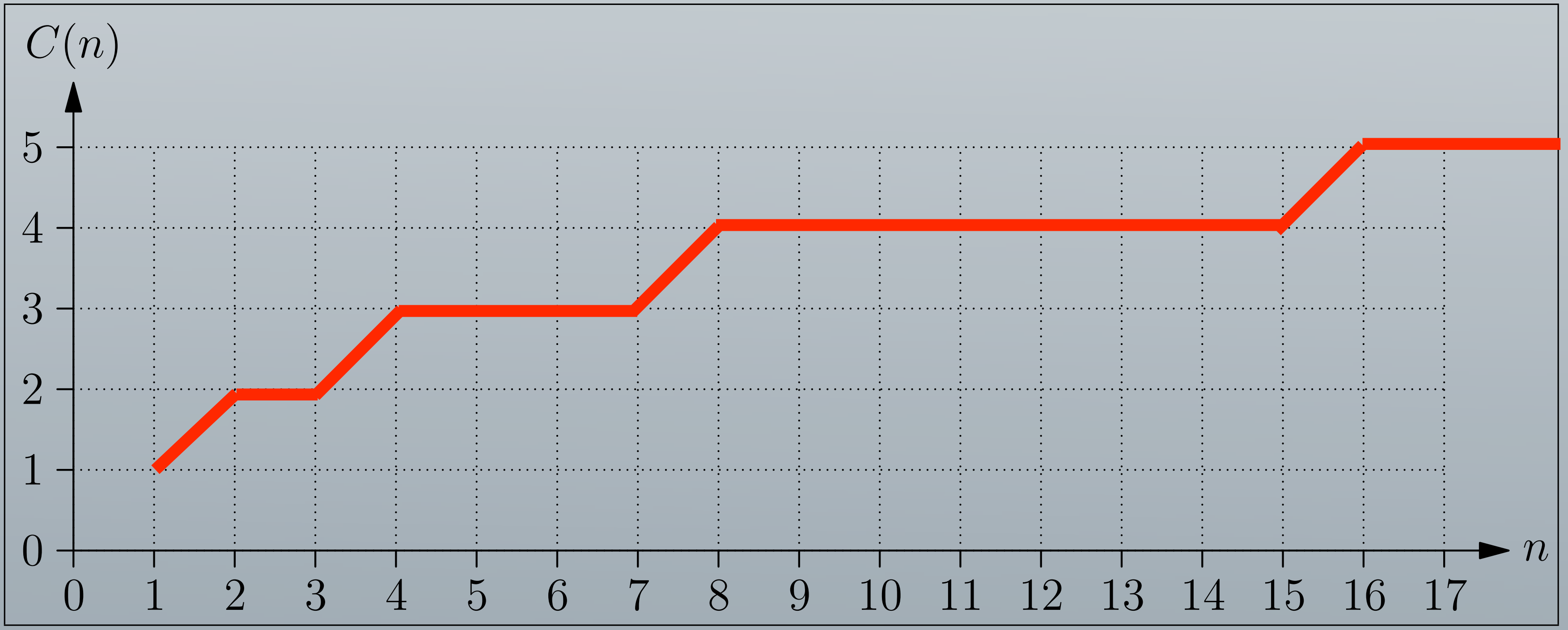
(a) Let $C(n)$ be the number of times the function `bar()` is called when we call `foo(n)`. Write down a recurrence relation for $C(n)$. *(2 marks)*

$C(n) =$ $C(\lfloor n/2 \rfloor) + 1$

(b) Write down the boundary condition for the recurrence relation. *(1 marks)*

$C(1) =$ 1

(c) Using the recurrence compute values of $C(n)$ to draw the graph below. *(4 marks)*



3 Consider the program (valid for inputs $n \geq 1$)

$$S_m = \{2^m, 2^m + 1, \ldots, 2^{m+1} - 1\}$$

}

where (int) n/2 returns the greatest integer less than or equal to n/2 (i.e. $\lfloor n/2 \rfloor$).

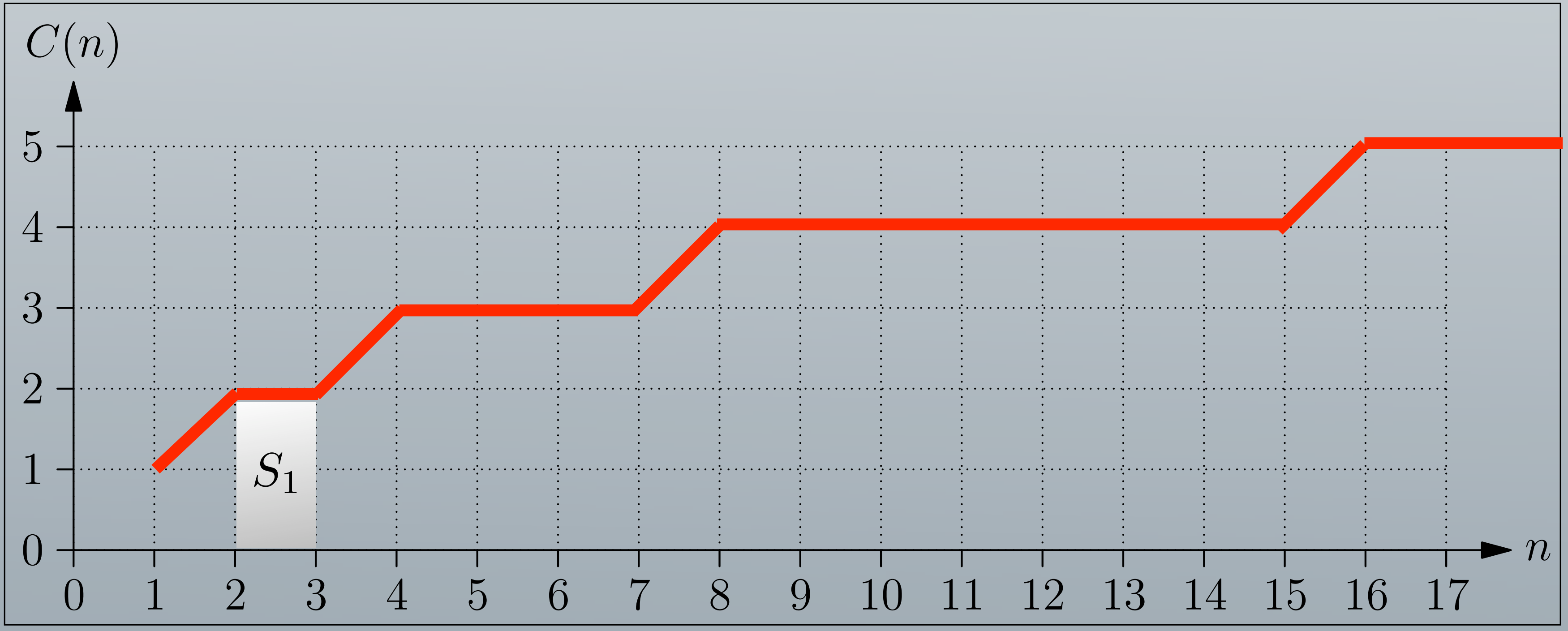
(a) Let $C(n)$ be the number of times the function `bar()` is called when we call `foo(n)`. Write down a recurrence relation for $C(n)$. *(2 marks)*

$C(n) =$ C(⌊n/2⌋) + 1

(b) Write down the boundary condition for the recurrence relation. *(1 marks)*

$C(1) =$ 1

(c) Using the recurrence compute values of $C(n)$ to draw the graph below. *(4 marks)*



3 Consider the program (valid for inputs $n \geq 1$)

$$S_m = \{2^m, 2^m + 1, \ldots, 2^{m+1} - 1\}$$

}

where `(int) n/2` returns the greatest integer less than or equal to n/2 (i.e. $\lfloor n/2 \rfloor$).

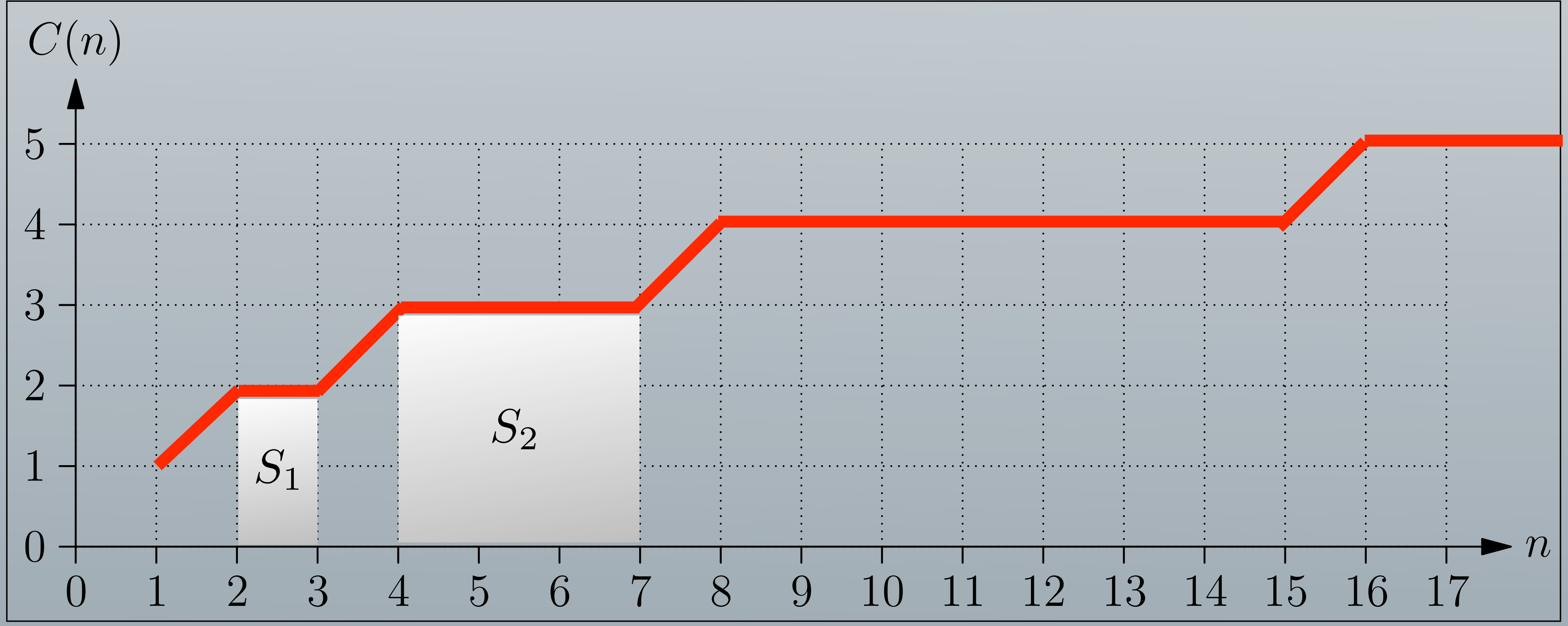
(a) Let $C(n)$ be the number of times the function `bar()` is called when we call `foo(n)`. Write down a recurrence relation for $C(n)$. *(2 marks)*

$C(n) =$ C(⌊n/2⌋) + 1

(b) Write down the boundary condition for the recurrence relation. *(1 marks)*

$C(1) =$ 1

(c) Using the recurrence compute values of $C(n)$ to draw the graph below. *(4 marks)*



3 Consider the program (valid for inputs $n \geq 1$)

$$S_m = \{2^m, 2^m + 1, \ldots, 2^{m+1} - 1\}$$

}

where `(int) n/2` returns the greatest integer less than or equal to n/2 (i.e. $\lfloor n/2 \rfloor$).

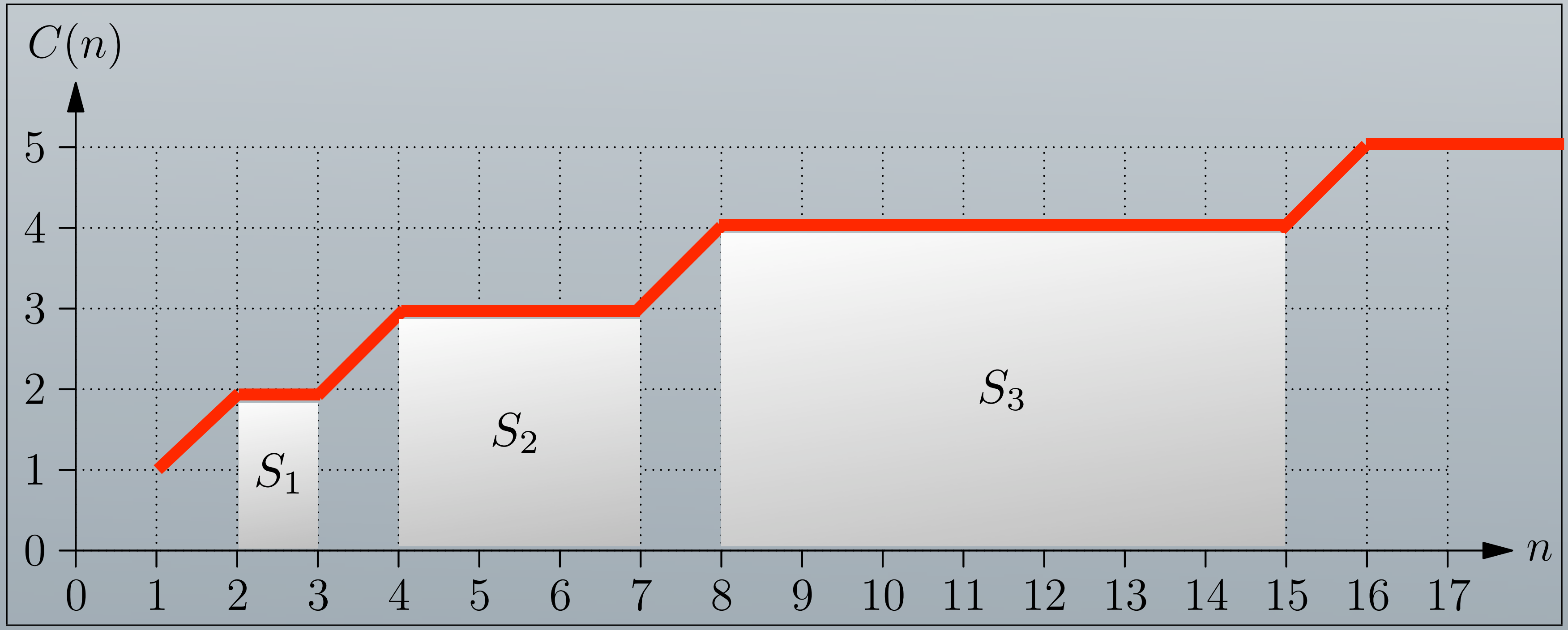
(a) Let $C(n)$ be the number of times the function `bar()` is called when we call `foo(n)`. Write down a recurrence relation for $C(n)$. *(2 marks)*

$C(n) =$ $C(\lfloor n/2 \rfloor) + 1$

(b) Write down the boundary condition for the recurrence relation. *(1 marks)*

$C(1) =$ 1

(c) Using the recurrence compute values of $C(n)$ to draw the graph below. *(4 marks)*



3 Consider the program (valid for inputs $n \geq 1$)

$$S_m = \{2^m, 2^m + 1, \ldots, 2^{m+1} - 1\}$$

whe

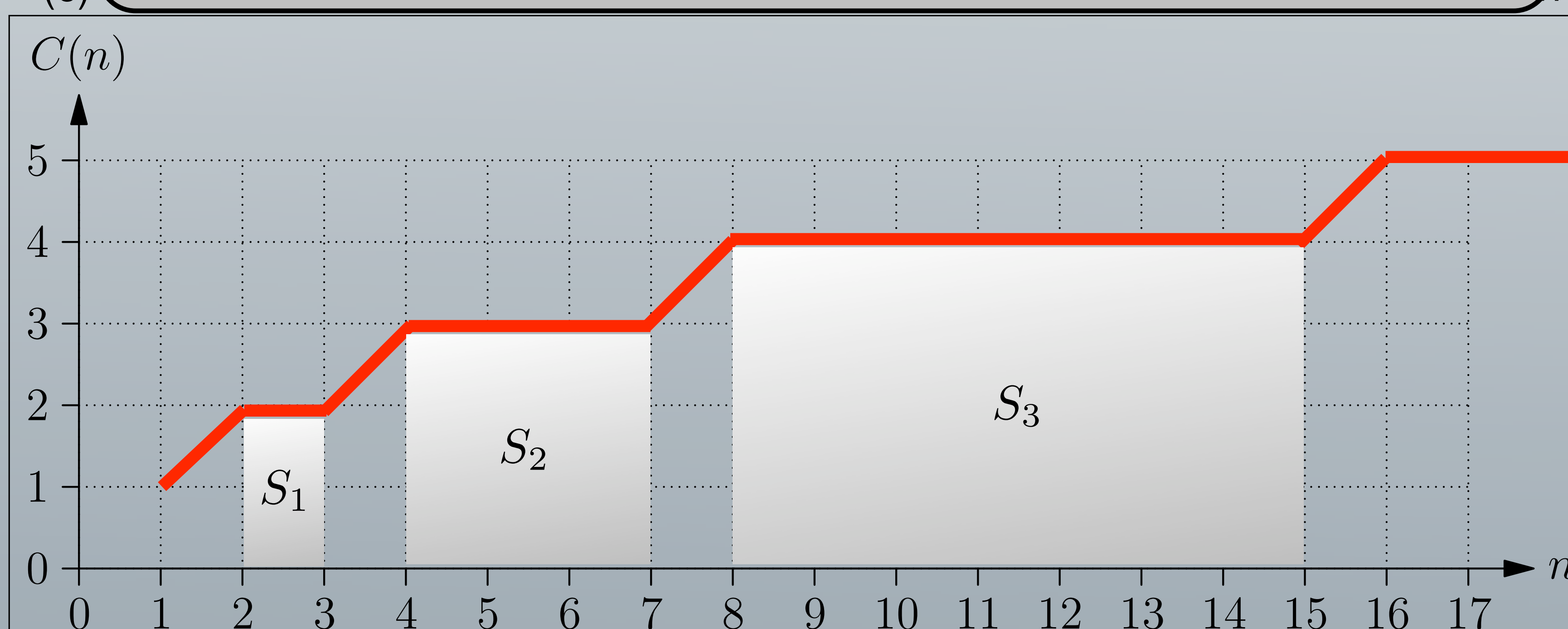
Notice that the value of C(n) is constant across all n in S_m

it's m+1

(a)

(b)

(c)



Question Three (d)

Claim: $f(n) = \lfloor \log_2(n) \rfloor + 1$ is a solution to

$$C(n) = C(\lfloor n/2 \rfloor) + 1$$
$$C(1) = 1$$

Question Three (d)

Claim: $f(n) = \lfloor \log_2(n) \rfloor + 1$ is a solution to

$$C(n) = C(\lfloor n/2 \rfloor) + 1$$
$$C(1) = 1$$

Proof:

We proceed by induction on m to show that C(n) = f(n) whenever n is in S_m

Question Three (d)

$$C(n) = C(\lfloor n/2 \rfloor) + 1$$
$$C(1) = 1$$

Claim: $f(n) = \lfloor \log_2(n) \rfloor + 1$ is a solution to

Proof:

We proceed by induction on m to show that C(n) = f(n) whenever n is in S_m

Base Case: (m=0)

Question Three (d)

Claim: $f(n) = \lfloor \log_2(n) \rfloor + 1$ is a solution to

$$C(n) = C(\lfloor n/2 \rfloor) + 1$$
$$C(1) = 1$$

Proof:

We proceed by induction on m to show that C(n) = f(n) whenever n is in S_m

Base Case: (m=0)

We know $S_0 = \{1\}$ so we must have n is 1

Question Three (d)

Claim: $f(n) = \lfloor \log_2(n) \rfloor + 1$ is a solution to $C(n) = C(\lfloor n/2 \rfloor) + 1$, $C(1) = 1$

Proof:

We proceed by induction on m to show that C(n) = f(n) whenever n is in S_m

Base Case: (m=0)

We know $S_0 = \{1\}$ so we must have n is 1

$$C(1) = 1$$

Question Three (d)

Claim: $f(n) = \lfloor \log_2(n) \rfloor + 1$ is a solution to

$$C(n) = C(\lfloor n/2 \rfloor) + 1$$
$$C(1) = 1$$

Proof:

We proceed by induction on m to show that C(n) = f(n) whenever n is in S_m

Base Case: (m=0)

We know $S_0 = \{1\}$ so we must have n is 1

$$C(1) = 1$$

$$f(1) = \lfloor \log_2(1) \rfloor + 1 = 1$$

Question Three (d)

Claim: $f(n) = \lfloor \log_2(n) \rfloor + 1$ is a solution to $\begin{array}{c} C(n) = C(\lfloor n/2 \rfloor) + 1 \\ C(1) = 1 \end{array}$

Proof:

We proceed by induction on m to show that C(n) = f(n) whenever n is in S_m

Base Case: (m=0)

We know $S_0 = \{1\}$ so we must have n is 1

$$C(1) = 1$$

$$f(1) = \lfloor \log_2(1) \rfloor + 1 = 1$$

Therefore $C(1) = f(1)$ as required.

Question Three (d)

Claim: $f(n) = \lfloor \log_2(n) \rfloor + 1$ **is a solution to**

$$C(n) = C(\lfloor n/2 \rfloor) + 1$$
$$C(1) = 1$$

Proof:

We proceed by induction on m to show that C(n) = f(n) whenever n is in S_m

Step Case: (m>0)

Question Three (d)

$$C(n) = C(\lfloor n/2 \rfloor) + 1$$
$$C(1) = 1$$

Claim: $f(n) = \lfloor \log_2(n) \rfloor + 1$ **is a solution to**

Proof:

We proceed by induction on m to show that C(n) = f(n) whenever n is in S_m

Step Case: (m>0)

Suppose that C(n) = f(n) for all n in $S_{m-1} = \{2^{m-1}, \ldots, 2^m - 1\}$

Question Three (d)

Claim: $f(n) = \lfloor \log_2(n) \rfloor + 1$ is a solution to

$$C(n) = C(\lfloor n/2 \rfloor) + 1$$
$$C(1) = 1$$

Proof:

We proceed by induction on m to show that C(n) = f(n) whenever n is in S_m

Step Case: (m>0)

Suppose that C(n) = f(n) for all n in $S_{m-1} = \{2^{m-1}, \ldots, 2^m - 1\}$

Now take any n in $S_m = \{2^m, \ldots, 2^m - 1\}$

Question Three (d)

Claim: $f(n) = \lfloor \log_2(n) \rfloor + 1$ is a solution to

$$C(n) = C(\lfloor n/2 \rfloor) + 1$$
$$C(1) = 1$$

Proof:

We proceed by induction on m to show that C(n) = f(n) whenever n is in S_m

Step Case: (m>0)

Suppose that C(n) = f(n) for all n in $S_{m-1} = \{2^{m-1}, \ldots, 2^m - 1\}$

Now take any n in $S_m = \{2^m, \ldots, 2^m - 1\}$

$C(n) = C(\lfloor n/2 \rfloor) + 1$ and notice that $\lfloor n/2 \rfloor$ is in S_{m-1}

Question Three (d)

$$C(n) = C(\lfloor n/2 \rfloor) + 1$$
$$C(1) = 1$$

Claim: $f(n) = \lfloor \log_2(n) \rfloor + 1$ is a solution to

Proof:

We proceed by induction on m to show that C(n) = f(n) whenever n is in S_m

Step Case: (m>0)

Suppose that C(n) = f(n) for all n in $S_{m-1} = \{2^{m-1}, \ldots, 2^m - 1\}$

Now take any n in $S_m = \{2^m, \ldots, 2^m - 1\}$

$$C(n) = C(\lfloor n/2 \rfloor) + 1$$ and notice that $\lfloor n/2 \rfloor$ is in S_{m-1}

$$= f(\lfloor n/2 \rfloor) + 1$$ by inductive hypothesis

Question Three (d)

$$C(n) = C(\lfloor n/2 \rfloor) + 1$$
$$C(1) = 1$$

Claim: $f(n) = \lfloor \log_2(n) \rfloor + 1$ is a solution to

Proof:

We proceed by induction on m to show that C(n) = f(n) whenever n is in S_m

Step Case: (m>0)

Suppose that C(n) = f(n) for all n in $S_{m-1} = \{2^{m-1}, \ldots, 2^m - 1\}$

Now take any n in $S_m = \{2^m, \ldots, 2^m - 1\}$

$C(n) = C(\lfloor n/2 \rfloor) + 1$ and notice that $\lfloor n/2 \rfloor$ is in S_{m-1}

$= f(\lfloor n/2 \rfloor) + 1$ by inductive hypothesis

Also notice that $f(n) = m + 1$ whenever n is in S_m

Question Three (d)

Claim: $f(n) = \lfloor \log_2(n) \rfloor + 1$ is a solution to

$$C(n) = C(\lfloor n/2 \rfloor) + 1$$
$$C(1) = 1$$

Proof:

We proceed by induction on m to show that (

Step Case: (m>0)

Suppose that C(n) = f(n) for all n in $S_{m-1} = \{2^{m-}$

Now take any n in $S_m = \{2^m, \ldots, 2^m - 1\}$

$$C(n) = C(\lfloor n/2 \rfloor) + 1$$ and notice tha

$$= f(\lfloor n/2 \rfloor) + 1$$ by inductive h

Also notice that $f(n) = m + 1$ whenever n is in S_m

Because :

$n \in \{2^m, \ldots 2^{m+1} - 1\}$

implies

$\lfloor \log_2(n) \rfloor = m$

so

$f(n) = \lfloor \log_2(n) \rfloor + 1 = m + 1$

Question Three (d)

Claim: $f(n) = \lfloor \log_2(n) \rfloor + 1$ is a solution to

$$C(n) = C(\lfloor n/2 \rfloor) + 1$$
$$C(1) = 1$$

Proof:

We proceed by induction on m to show that C(n) = f(n) whenever n is in S_m

Step Case: (m>0)

Suppose that C(n) = f(n) for all n in $S_{m-1} = \{2^{m-1}, \ldots, 2^m - 1\}$

Now take any n in $S_m = \{2^m, \ldots, 2^m - 1\}$

$$C(n) = C(\lfloor n/2 \rfloor) + 1$$ and notice that $\lfloor n/2 \rfloor$ is in S_{m-1}

$$= f(\lfloor n/2 \rfloor) + 1$$ by inductive hypothesis

Also notice that $f(n) = m + 1$ whenever n is in S_m

Question Three (d)

Claim: $f(n) = \lfloor \log_2(n) \rfloor + 1$ **is a solution to**
$$C(n) = C(\lfloor n/2 \rfloor) + 1$$
$$C(1) = 1$$

Proof:

We proceed by induction on m to show that C(n) = f(n) whenever n is in S_m

Step Case: (m>0)

Suppose that C(n) = f(n) for all n in $S_{m-1} = \{2^{m-1}, \ldots, 2^m - 1\}$

Now take any n in $S_m = \{2^m, \ldots, 2^m - 1\}$

$$C(n) = C(\lfloor n/2 \rfloor) + 1$$ and notice that $\lfloor n/2 \rfloor$ is in S_{m-1}

$$= f(\lfloor n/2 \rfloor) + 1$$ by inductive hypothesis

Also notice that $f(n) = m + 1$ whenever n is in S_m

Therefore $f(\lfloor n/2 \rfloor) = (m - 1) + 1$

Question Three (d)

Claim: $f(n) = \lfloor \log_2(n) \rfloor + 1$ is a solution to $$C(n) = C(\lfloor n/2 \rfloor) + 1$$ $$C(1) = 1$$

Proof:

We proceed by induction on m to show that C(n) = f(n) whenever n is in S_m

Step Case: (m>0)

Suppose that C(n) = f(n) for all n in $S_{m-1} = \{2^{m-1}, \ldots, 2^m - 1\}$

Now take any n in $S_m = \{2^m, \ldots, 2^m - 1\}$

$$C(n) = C(\lfloor n/2 \rfloor) + 1$$ and notice that $\lfloor n/2 \rfloor$ is in S_{m-1}

$$= f(\lfloor n/2 \rfloor) + 1$$ by inductive hypothesis

Also notice that $f(n) = m + 1$ whenever n is in S_m

Therefore $f(\lfloor n/2 \rfloor) = (m - 1) + 1$ which tells us

$$C(n) = ((m - 1) + 1) + 1$$

Question Three (d)

Claim: $f(n) = \lfloor \log_2(n) \rfloor + 1$ **is a solution to** $$C(n) = C(\lfloor n/2 \rfloor) + 1$$ $$C(1) = 1$$

Proof:

We proceed by induction on m to show that C(n) = f(n) whenever n is in S_m

Step Case: (m>0)

Suppose that C(n) = f(n) for all n in $S_{m-1} = \{2^{m-1}, \ldots, 2^m - 1\}$

Now take any n in $S_m = \{2^m, \ldots, 2^m - 1\}$

$$C(n) = C(\lfloor n/2 \rfloor) + 1$$ and notice that $\lfloor n/2 \rfloor$ is in S_{m-1}

$$= \boxed{f(\lfloor n/2 \rfloor)} + 1$$ by inductive hypothesis

Also notice that $f(n) = m + 1$ whenever n is in S_m

Therefore $f(\lfloor n/2 \rfloor) = (m-1) + 1$ which tells us

$$C(n) = \boxed{((m-1)+1)} + 1$$

Question Three (d)

Claim: $f(n) = \lfloor \log_2(n) \rfloor + 1$ is a solution to $$C(n) = C(\lfloor n/2 \rfloor) + 1$$ $$C(1) = 1$$

Proof:

We proceed by induction on m to show that C(n) = f(n) whenever n is in S_m

Step Case: (m>0)

Suppose that C(n) = f(n) for all n in $S_{m-1} = \{2^{m-1}, \ldots, 2^m - 1\}$

Now take any n in $S_m = \{2^m, \ldots, 2^m - 1\}$

$C(n) = C(\lfloor n/2 \rfloor) + 1$ and notice that $\lfloor n/2 \rfloor$ is in S_{m-1}

$= f(\lfloor n/2 \rfloor) + 1$ by inductive hypothesis

Also notice that $f(n) = m + 1$ whenever n is in S_m

Therefore $f(\lfloor n/2 \rfloor) = (m-1) + 1$ which tells us

$$C(n) = ((m-1) + 1) + 1$$

Question Three (d)

Claim: $f(n) = \lfloor \log_2(n) \rfloor + 1$ is a solution to
$$C(n) = C(\lfloor n/2 \rfloor) + 1$$
$$C(1) = 1$$

Proof:

We proceed by induction on m to show that C(n) = f(n) whenever n is in S_m

Step Case: (m>0)

Suppose that C(n) = f(n) for all n in $S_{m-1} = \{2^{m-1}, \ldots, 2^m - 1\}$

Now take any n in $S_m = \{2^m, \ldots, 2^m - 1\}$

$$C(n) = C(\lfloor n/2 \rfloor) + 1$$ and notice that $\lfloor n/2 \rfloor$ is in S_{m-1}

$$= f(\lfloor n/2 \rfloor) + 1$$ by inductive hypothesis

Also notice that $f(n) = m + 1$ whenever n is in S_m

Therefore $f(\lfloor n/2 \rfloor) = (m-1) + 1$ which tells us

$$C(n) = ((m-1)+1)+1$$

$$= m + 1 = f(n)$$ as required.

(d) Prove by induction that $f(n) = \lfloor \log_2(n) \rfloor + 1$ satisfies the recurrence relation for $C(n)$. This is simplified if we perform the inductive step over the set of integers $S_m = \{2^m, 2^m + 1, \ldots, 2^{m+1} - 1\}$. *(6 marks)*

(e) If it takes 100s to compute `foo(512)` approximately how long will it take to compute `foo(1024)`? *(2 marks)*

Question Three (e)

Time to run foo(n) is approximately $c(\log_2(n) + 1)$

Question Three (e)

Time to run foo(n) is approximately $c(\log_2(n) + 1)$

We know $T(foo(512)) = 100 \approx c \times (\log_2(512) + 1) \approx 10c$

Question Three (e)

Time to run foo(n) is approximately $c(\log_2(n) + 1)$

We know $T(foo(512)) = 100 \approx c \times (\log_2(512) + 1) \approx 10c$

Therefore $c \approx 10$

Question Three (e)

Time to run foo(n) is approximately $c(\log_2(n) + 1)$

We know $T(foo(512)) = 100 \approx c \times (\log_2(512) + 1) \approx 10c$

Therefore $c \approx 10$

Now $T(foo(1024)) \approx c \times (\log_2(1024) + 1)$

Question Three (e)

Time to run foo(n) is approximately $c(\log_2(n) + 1)$

We know $T(foo(512)) = 100 \approx c \times (\log_2(512) + 1) \approx 10c$

Therefore $c \approx 10$

Now $T(foo(1024)) \approx c \times (\log_2(1024) + 1)$

$$\approx 10 \times 11$$

Question Three (e)

Time to run foo(n) is approximately $c(\log_2(n) + 1)$

We know $T(foo(512)) = 100 \approx c \times (\log_2(512) + 1) \approx 10c$

Therefore $c \approx 10$

Now $$\begin{aligned} T(foo(1024)) &\approx c \times (\log_2(1024) + 1) \\ &\approx 10 \times 11 \\ &= 110s \end{aligned}$$

That's all folks.