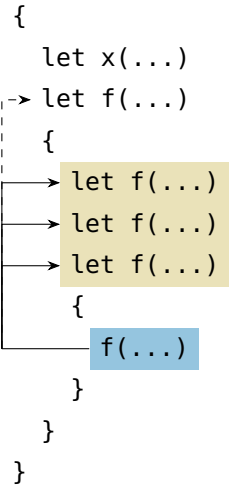
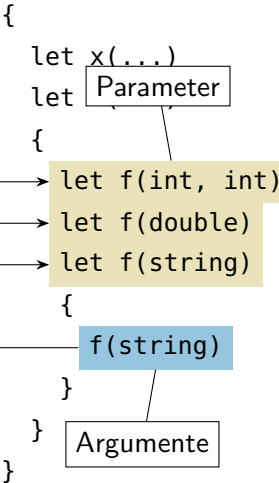


```
{  
  let x(...)  
  let f(...)  
  {  
    let f(...)  
    let f(...)  
    let f(...)  
    {  
      f(...)  
    }  
  }  
}
```



```
{
  let x(...)
  let f(...)
  {
    let f(...)
    let f(...)
    let f(...)
    {
      f(...)
    }
  }
}
```



```
{  
  let x(...)  
  let f(...)  
  {  
    let f(int, int)  
    let f(double)  
    let f(string)  
    {  
      f(string)  
    }  
  }  
}
```

The diagram illustrates function overloading in a programming language. It shows a function `f` being defined multiple times with different parameter lists. The lines `let f(double)` and `let f(string)` are highlighted in a yellow box, and the line `f(string)` is highlighted in a blue box. Arrows point from the left to these highlighted lines, indicating that they are the specific function definitions being called or referenced.

```
{
  let x(...)
  let f(...)
  {
    let f(int, int)
    let f(double)
    let f(string)
    {
      f(string)
    }
  }
}
```

The diagram illustrates a recursive call. A horizontal arrow points from the right side of the `f(string)` definition to the left side of the `f(string)` call. A vertical line descends from the left side of the call, then turns right to point at the `f(string)` definition.