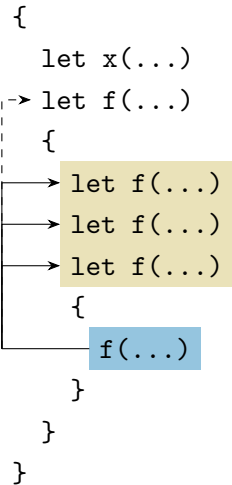
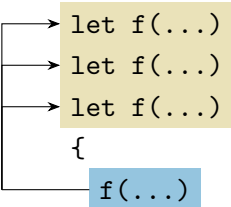


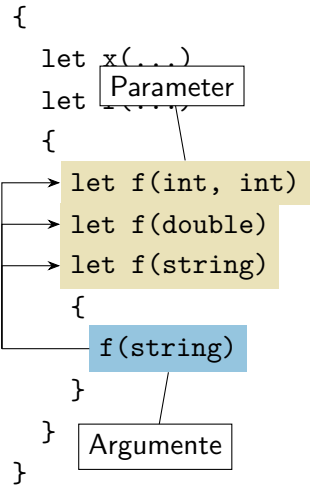
```
{  
  let x(...)  
  let f(...)  
  {  
    let f(...)  
    let f(...)  
    let f(...)  
    {  
      f(...)  
    }  
  }  
}
```



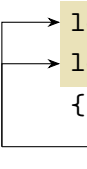
```
{  
  let x(...)  
  let f(...)  
  {  
    let f(...)  
    let f(...)  
    let f(...)  
    {  
      f(...)  
    }  
  }  
}
```



The diagram illustrates the scope resolution for the variable `f`. Three arrows originate from the left and point to the three `let f(...)` statements within the inner block. This indicates that each of these statements is being processed, likely to determine the final binding of `f` within that scope.



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



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

The diagram illustrates the resolution of a function call `f(string)` within a nested scope. A line originates from the `f(string)` call in the innermost block and points to the `let f(string)` definition in the same block, indicating that the call is resolved to the local function definition.