Given PLAN:
STEP 1: Define a function maxProfit which takes an array of integers as input and produces an integer as output.
STEP 2: In the main function, let the "prices" variable hold the input array of integers [7, 1, 5, 3, 6, 4].
STEP 3: In the main function, let the "result" variable hold the output of maxProfit given the input array of integers.
STEP 4: In the maxProfit function, initiate the conditional checks with parameters prices, (maxBound :: Int), and 0.
STEP 5: Define two checks. Check 1 verifies whether the first parameter is an empty list and the second parameter is the maximum bound of the integer type. Check 2 verifies whether the first parameter is a non-empty list. These checks are executed in the order in which they appear. In the case of the [7, 1, 5, 3, 6, 4] input array, given the ordering of the checks, only check 2 is executed.
STEP 6: In the maxProfit function, define a helper function maxProfitHelper which takes three parameters: an array of integers, an integer representing the minimum price, and an integer representing the maximum profit. This helper function recursively calculates the maximum profit by updating the minimum price and maximum profit values based on the current element of the input array.
STEP 7: In the main function, print the resulting maximum profit to the screen.

So the CODE COVERAGE for the given code snippet will be:

```
> maxProfit :: [Int] -> Int
> maxProfit prices = maxProfitHelper prices (maxBound :: Int) 0
>   where
>     maxProfitHelper [] _ maxProfit = maxProfit
>     maxProfitHelper (p:ps) minPrice maxProfit =
>       let newMinPrice = min minPrice p
>           profit = p - newMinPrice
>           newMaxProfit = max maxProfit profit
>       in maxProfitHelper ps newMinPrice newMaxProfit
> main = do
>     let prices = [7, 1, 5, 3, 6, 4]
>     let result = maxProfit prices
>     putStrLn $ "Maximum profit: " ++ show result
```