Danger and Target Movement

Directions: Use the Design Recipe to write a function `update-danger`, which takes in the danger's x-coordinate and produces the next x-coordinate.

Contract and Purpose Statement

Every contract has three parts…

```
# update-danger:: Number -> Number
```
function name *domain* *range*

```
# Consumes an x-coordinate and returns a new x-coordinate
```
what does the function do?

Examples

Write some examples, then circle and label what changes…

examples:

```
update-danger ( 160 ) is 160 - 50
```
function name *input(s)* *what the function produces*

```
update-danger ( -85 ) is -85 - 50
```
function name *input(s)* *what the function produces*

end

Definition

Write the definition, giving variable names to all your input values…

```
fun update-danger( x ):
```
function name *variable(s)*

```
  x - 50
```
what the function does with those variable(s)

end

Directions: Use the Design Recipe to write a function `update-target`, which takes in the danger's x-coordinate and produces the next x-coordinate.

Contract and Purpose Statement

Every contract has three parts…

```
# update-target:: Number -> Number
```
function name *domain* *range*

```
# Consumes an x-coordinate and returns a new x-coordinate
```
what does the function do?

Examples

Write some examples, then circle and label what changes…

examples:

```
update-target ( 130 ) is 130 + 50
```
function name *input(s)* *what the function produces*

```
update-target ( -25 ) is -25 + 50
```
function name *input(s)* *what the function produces*

end

Definition

Write the definition, giving variable names to all your input values…

```
fun update-target( x ):
```
function name *variable(s)*

```
  x + 50
```
what the function does with those variable(s)

end