

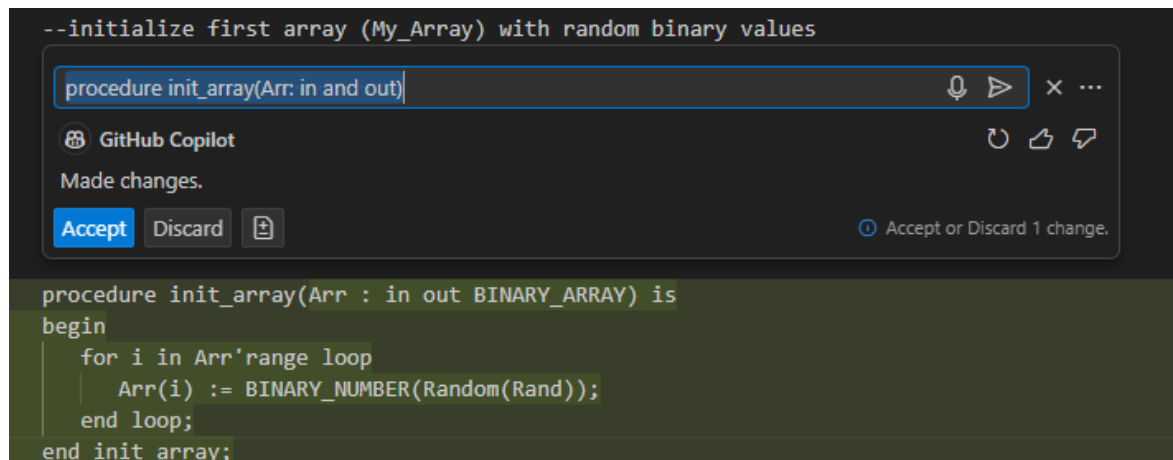
Clay Ramey

Professor Sardinas

Comp-3220 | HW-7

Given the five functions we were asked to create, I used copilot on two. The first function was to create a first array with random binary numbers.

Below is the output and recommendation from GitHub Copilot:



The screenshot shows a code editor with a dark theme. At the top, a comment reads: `--initialize first array (My_Array) with random binary values`. Below this, a GitHub Copilot suggestion is displayed in a light blue box. The suggestion contains the following PLI code:

```
procedure init_array(Arr: in out BINARY_ARRAY) is
begin
  for i in Arr'range loop
    Arr(i) := BINARY_NUMBER(Random(Rand));
  end loop;
end init_array;
```

Below the code suggestion, there is a status bar that says "Made changes." and buttons for "Accept", "Discard", and a plus icon. To the right of these buttons, it says "Accept or Discard 1 change."

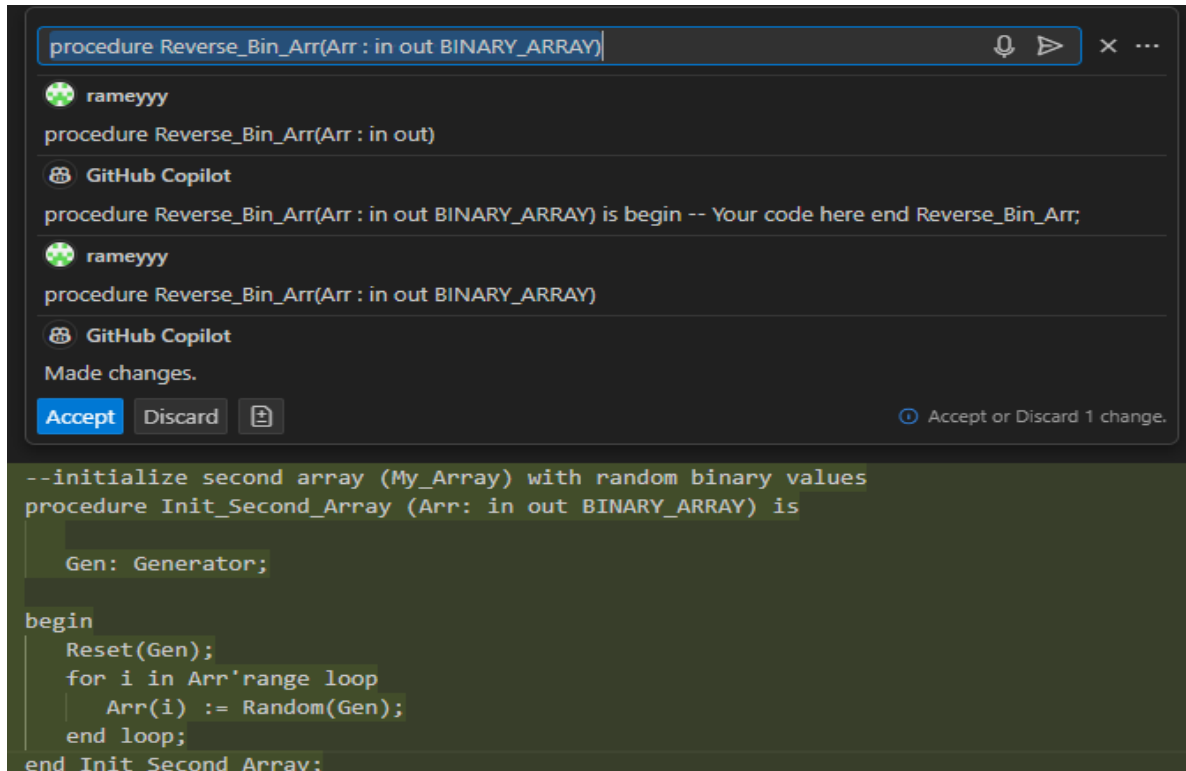
I took the for loop implementation with `Arr(i)`, and just added a reset call and took out the 'BINARY_NUMBER' function call, leaving my function like this:

```
--initialize first array (My_Array) with random binary values
procedure Init_Array (Arr: in out BINARY_ARRAY) is
  Gen: Generator;

begin
  Reset(Gen);
  for i in Arr'range loop
    Arr(i) := Random(Gen);
  end loop;
end Init Array;
```

The second function I used Copilot for was Reverse_Bin_Arr, with the goal to reverse the binary array.

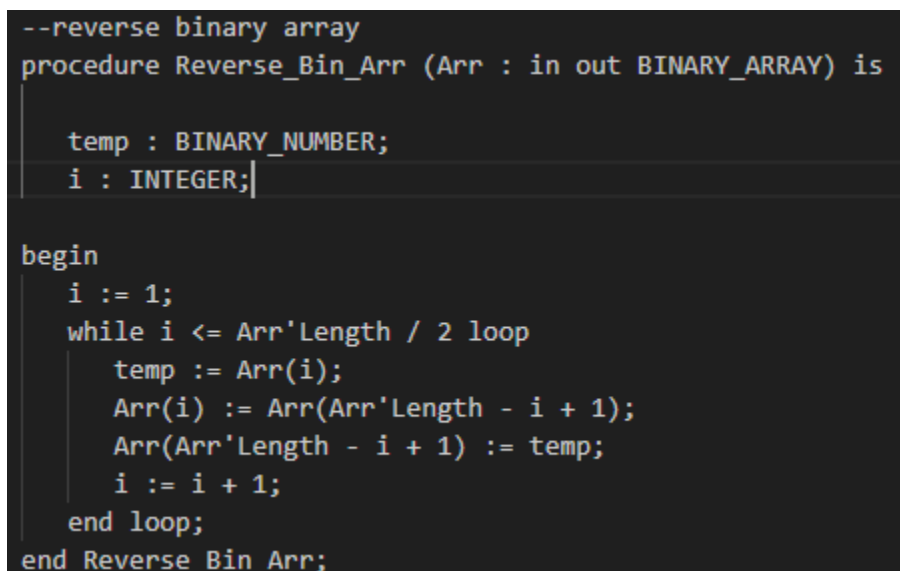
GitHub Copilot gave me this recommendation for the function:



The screenshot shows a code editor with a Copilot suggestion. The suggestion is for a procedure named `Reverse_Bin_Arr` that takes an `Arr` of type `BINARY_ARRAY` as input and output. The suggestion includes a comment `--initialize second array (My_Array) with random binary values` and a procedure `Init_Second_Array` that uses a `Generator` to fill the array with random values. The suggestion is presented in a dark-themed window with a search bar at the top and a status bar at the bottom.

```
procedure Reverse_Bin_Arr(Arr : in out BINARY_ARRAY) is
begin
  --initialize second array (My_Array) with random binary values
  procedure Init_Second_Array (Arr : in out BINARY_ARRAY) is
  Gen: Generator;
  begin
    Reset(Gen);
    for i in Arr'range loop
      Arr(i) := Random(Gen);
    end loop;
  end Init_Second_Array;
end Reverse_Bin_Arr;
```

It didn't understand my request and created a second array function. I changed it to a reversed insertion sort to do what the function was intended to do:



The screenshot shows a code editor with a PLI procedure `Reverse_Bin_Arr` that takes an `Arr` of type `BINARY_ARRAY` as input and output. The procedure uses a `temp` variable of type `BINARY_NUMBER` and an `i` variable of type `INTEGER` to reverse the array. The procedure is presented in a dark-themed window with a search bar at the top and a status bar at the bottom.

```
--reverse binary array
procedure Reverse_Bin_Arr (Arr : in out BINARY_ARRAY) is
temp : BINARY_NUMBER;
i : INTEGER;

begin
  i := 1;
  while i <= Arr'Length / 2 loop
    temp := Arr(i);
    Arr(i) := Arr(Arr'Length - i + 1);
    Arr(Arr'Length - i + 1) := temp;
    i := i + 1;
  end loop;
end Reverse_Bin_Arr;
```