Simple Navigation (INDIVIDUAL) [SOLUTION]

1 pt – comments somewhere

```
with Ada.Text_Io;
with Ada.Integer_Text_IO;
use Ada.Text_Io;
use Ada.Integer_Text_Io;


procedure nav is
        --Constant for number of squares
        num_squares: constant Integer := 100;

        --Array for storing bad positions
        good_positions: array(1..num_squares) of Boolean;

        --Orientation Constants
        NORTH: constant Integer := 0;
        WEST:  constant Integer := 1;
        SOUTH: constant Integer := 2;
        EAST:  constant Integer := 3;

        --Direction Constants
        STRAIGHT: constant Integer := 0;
        LEFT:     constant Integer := 1;
        BACK:     constant Integer := 2;
        RIGHT:    constant Integer := 3;


   --Variables for integer input
        position: Integer;
        orientation : Integer;
        direction: Integer;

        --Variable for marking legality
        is_legal : Boolean;

begin

        --Initialize array
        for i in 1..100 loop
                good_positions(i) := true;
        end loop;
```

1 pt – creating the array

```
--Collect the invalid positions
Put("Enter the positions that it is invalid to travel to.  Enter 0 to indicate that you are done entering values:");

loop
        Get(Item => position);
        exit when position = 0;
        good_positions(position) := false;
end loop;

--Loop to collect desired inputs and outputs
loop
        Put("Enter current position (1-100), orientation (N=0,W=1,S=2,E=3), and desired direction (Straight=0,Left=1,Back=2,Right=3).");
        Put("Or enter 0 to exit.");
        New_Line;
        Get(Item => position);
        exit when position not in 1..100;

        Get(Item => orientation);
        Get(Item => direction);

        is_Legal := false;

        -- The constants for north, south, east, west and straight, left, back, right are aligned such
        -- that this calculation produces that cardinal direction that we are trying to go to.
        case ((direction + orientation) mod 4) is
                when NORTH =>
                        if position <= 10 then
                                -- this would move off the edge of the grid to the north
                                is_Legal := false;
                        else
                                is_Legal := good_positions(position-10);
                        end if;
                when SOUTH =>
                        if position >= 90 then
                                -- this would move off the edge of the grid to the south
                                is_Legal := false;
                        else
                                is_Legal := good_positions(position+10);
                        end if;
                when EAST =>
                        if (position mod 10) = 0 then
                                -- this would move off the edge of the grid to the east
                                is_Legal := false;
                        else
```

1 pt – get illegal positions from user

1 pt – a way for the user to exit the loop

1 pt – get illegal positions from user

8 pts – a control statement (decision structure) for checking for legal moves.

1 pt for checking if the rover moves off the grid in each direction.

1 pt for checking if the rover moves to an illegal square in each direction.

```
                                        is_Legal := good_positions(position+1);
                                end if;
                        when WEST =>
                                if (position mod 10) = 1 then
                                        -- this would move off the edge of the grid to the west
                                        is_Legal := false;
                                else
                                        is_Legal := good_positions(position-1);
                                end if;
                        when OTHERS =>
                                Put("Error: Invalid orientation");
                                exit;
                end case;

                if is_Legal then
                        Put("That move is legal.");
                else
                        Put("That move is not legal.");
                end if;
                New_Line;
                New_Line;

        end loop;

end nav;
```

1 pt – print something when a move is illegal

For a more intuitive, though less elegant, implementation you can replace the contents of the larger loop with:

```
loop
        Put("Enter current position (1-100), orientation (N=0,W=1,S=2,E=3), and desired direction (Straight=0,Left=1,Back=2,Right=3).");
        Put("Or enter 0 to exit.");
        New_Line;
        Get(Item => position);
        exit when position not in 1..100;

        Get(Item => orientation);
        Get(Item => direction);

        is_Legal := false;

        if(orientation=NORTH and direction=STRAIGHT) or
          (orientation=SOUTH and direction=BACK) or
          (orientation=WEST and direction=RIGHT) or
          (orientation=EAST and direction=LEFT) then
          --check the legality of moving north
                    if position <= 10 then
                              -- this would move off the edge of the grid to the north
                              is_Legal := false;
                    else
                              is_Legal := good_positions(position-10);
                    end if;
        elsif(orientation=NORTH and direction=BACK) or
          (orientation=SOUTH and direction=STRAIGHT) or
          (orientation=WEST and direction=LEFT) or
          (orientation=EAST and direction=RIGHT) then
          --check the legality of moving south
                    if position >= 90 then
                              -- this would move off the edge of the grid to the south
                              is_Legal := false;
                    else
                              is_Legal := good_positions(position+10);
                    end if;
    elsif(orientation=NORTH and direction=RIGHT) or
          (orientation=SOUTH and direction=LEFT) or
          (orientation=WEST and direction=BACK) or
          (orientation=EAST and direction=STRAIGHT) then
          --check the legality of moving east
                    if (position mod 10) = 0 then
                              -- this would move off the edge of the grid to the east
                              is_Legal := false;
```

```
                        else
                                is_Legal := good_positions(position+1);
                        end if;
                else
                  --check the legality of moving west
                        if (position mod 10) = 1 then
                                -- this would move off the edge of the grid to the west
                                is_Legal := false;
                        else
                                is_Legal := good_positions(position-1);
                        end if;
                end if;

                if is_Legal then
                        Put("That move is legal.");
                else
                        Put("That move is not legal.");
                end if;
                New_Line;
                New_Line;

        end loop;
```