COMP500 / ENSE501: Week 7 – Exercise:

EXERCISE NAME: *Number Guessing Game*

This program is a number guessing game where a simple AI opponent generates a secret number, between 1 and 23 inclusive, for the user to guess. The user must be able to take turns guessing the secret number, with the AI responding to the user's guesses. After successfully guessing, the user must be allowed to play again. The AI must count how many turns the user has taken.

Here is an example of a game in progress where the human user has incorrectly guessed the number `20` on their first turn:

```
BEEP, BUZZ, DING... Starting up the P1_AI_BOT...

P1BOT: Welcome to "The 'Basic' Number Guessing Game"!
P1BOT:
P1BOT: 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18;19,20,21,22,23
P1BOT: I'm thinking of a secret number, can you guess it...
P1BOT: Turn 1, what is your guess?

Human: 20

P1BOT: Wrong... my secret number comes before 20...
P1BOT: Turn 2, what is your guess?

Human:
```

When the program starts, it must print the following:

```
BEEP, BUZZ, DING... Starting up the P1_AI_BOT...

P1BOT: Welcome to "The 'Basic' Number Guessing Game"!
P1BOT:
```

After this, the AI must print the following messages:

```
P1BOT: 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18;19,20,21,22,23
P1BOT: I'm thinking of a secret number, can you guess it...
```

Carefully review the first line of the simple AI's message output above, it contains the following:

```
1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18;19,20,21,22,23
```

This simple AI is somewhat poor, as when it outputs the possible numbers, it also highlights what its randomly chosen secret number is by placing a semicolon after it instead of a comma! Notice the number `18`! A keen player may notice this (and a programmer can also use this knowledge when testing their implementation).

Next, the game must begin. The simple AI will prompt the user with the current turn number and ask the user to input their guess.

```
P1BOT: Turn 1, what is your guess?

Human: 20
```

Ensure the program validates user input to ensure an integer number is input.

When the user's response is wrong, the AI must respond appropriately. If the guess comes before the secret number, the simple AI must respond as follows:

```
P1BOT: Wrong... my secret number comes after 20...
```

If the guess comes after the secret number, the AI must respond as follows:

```
P1BOT: Wrong... my secret number comes before 20...
```

After this, the user must then be prompted to take their next turn, and make another guess:

```
P1BOT: Turn 2, what is your guess?

Human:
```

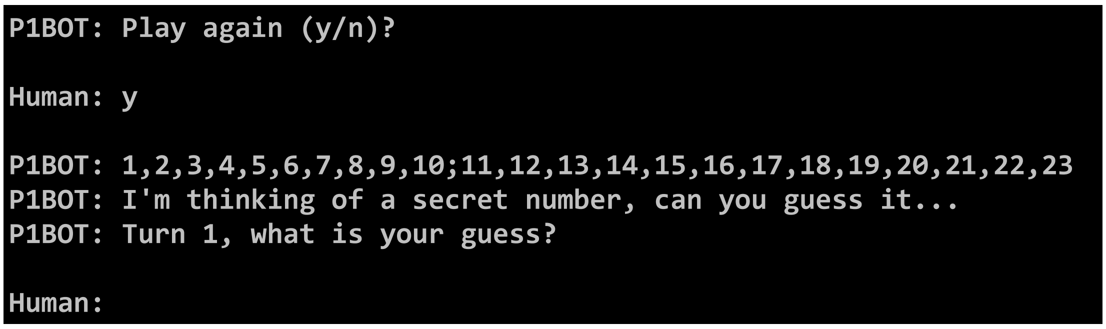
This pattern of play must continue until the user guesses correctly.

When the user guesses correctly, the AI must respond as follows and prompt the user whether they want to play again, or not, in the following style:

```
Human: 18

P1BOT: Well done!
P1BOT: You took 12 turns! Game over!
P1BOT: Play again (y/n)?
```

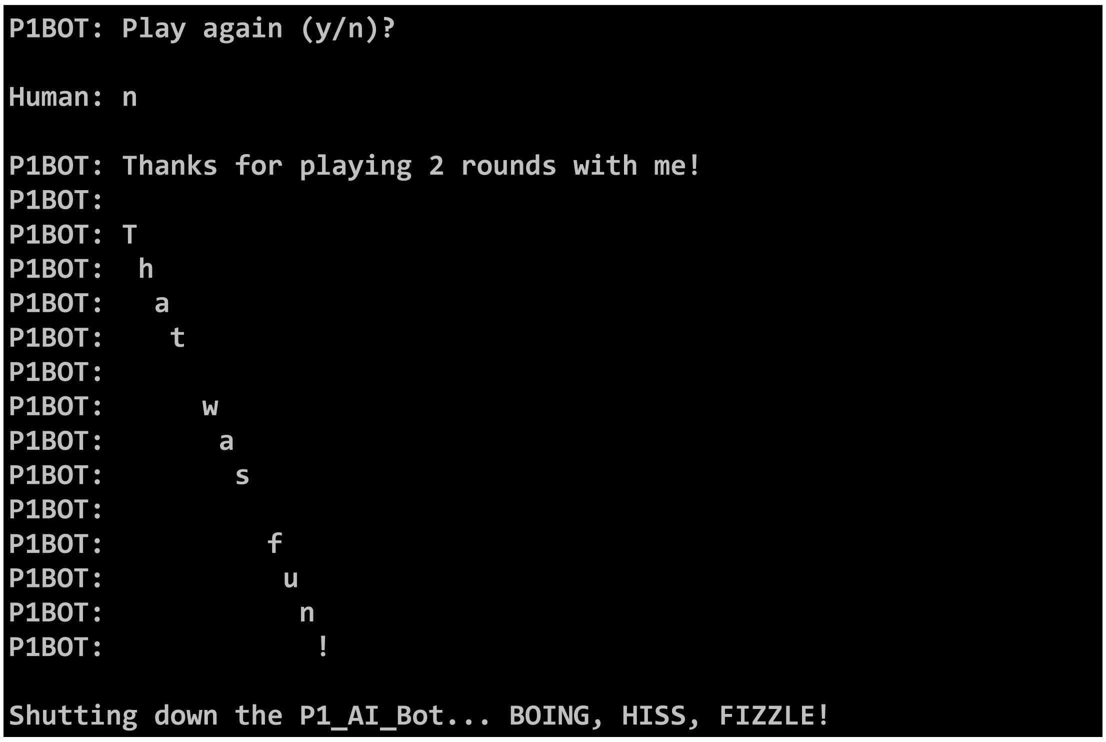
Ensure the "You took" some number of "turns!" is sensitive to plurals, and appropriately includes or excludes the **`'s'`** on the word "turns" depending on the number of turns taken!

When the user selects “yes” they want to play again, a new round must start:



```
P1BOT: Play again (y/n)?

Human: y

P1BOT: 1,2,3,4,5,6,7,8,9,10;11,12,13,14,15,16,17,18,19,20,21,22,23
P1BOT: I'm thinking of a secret number, can you guess it...
P1BOT: Turn 1, what is your guess?

Human:
```

When the user selects “no”, the following message is displayed before the program exits:

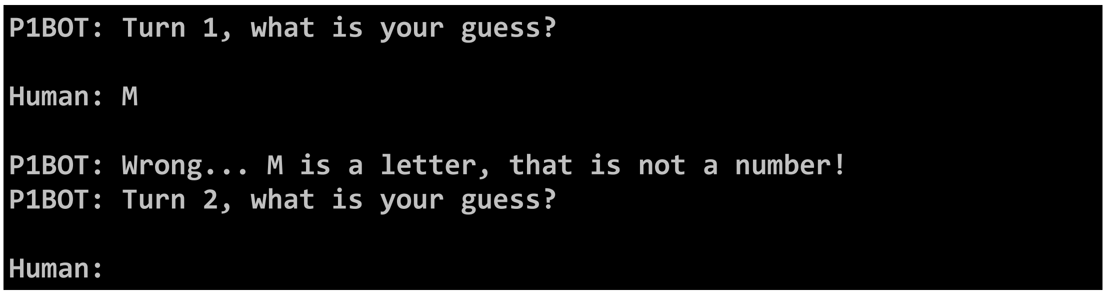


```
P1BOT: Play again (y/n)?

Human: n

P1BOT: Thanks for playing 2 rounds with me!
P1BOT:
P1BOT: T
P1BOT:  h
P1BOT:   a
P1BOT:    t
P1BOT:
P1BOT:      w
P1BOT:       a
P1BOT:        s
P1BOT:
P1BOT:          f
P1BOT:           u
P1BOT:            n
P1BOT:             !

Shutting down the P1_AI_Bot... BOING, HISS, FIZZLE!
```

Ensure the final “Thanks for playing” some number of “rounds with me” is sensitive to plurals, and appropriately includes or excludes the `'s'` on the word “rounds”!

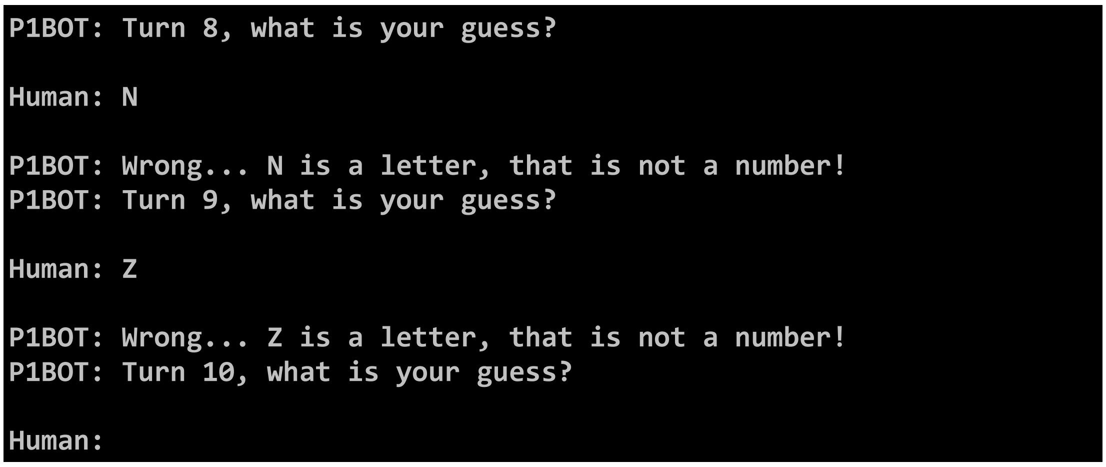
The program must also validate the user’s input.

When the user inputs a letter, the program must respond as follows:



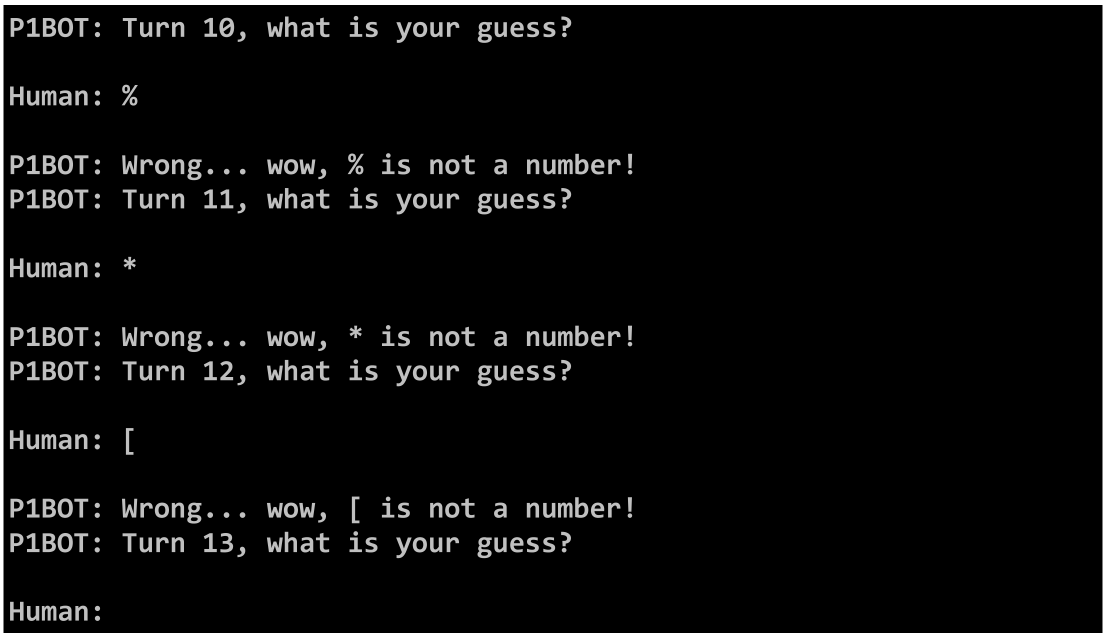
```
P1BOT: Turn 1, what is your guess?

Human: M

P1BOT: Wrong... M is a letter, that is not a number!
P1BOT: Turn 2, what is your guess?

Human:
```

Another example is as follows:



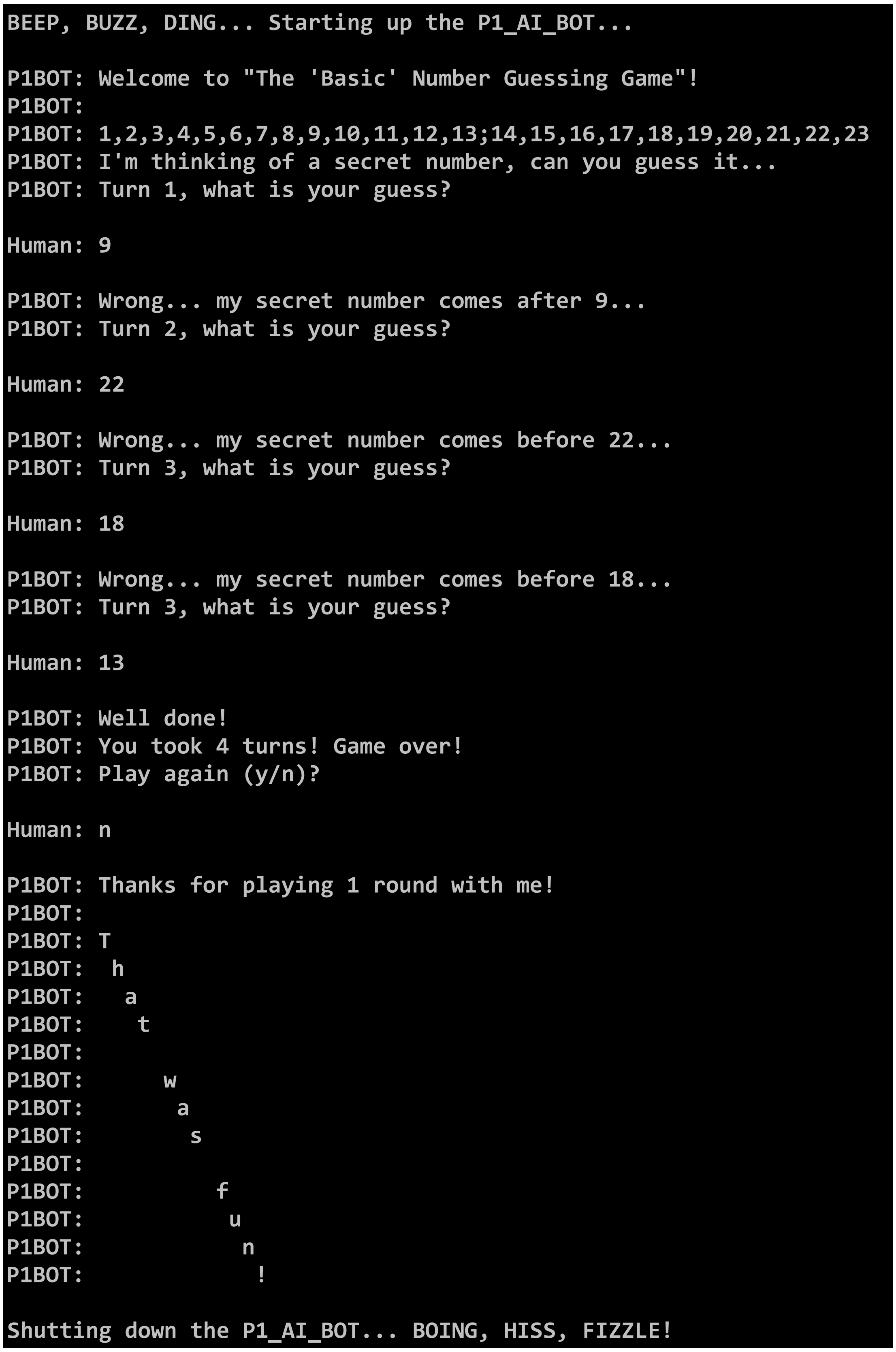
```
P1BOT: Turn 8, what is your guess?

Human: N

P1BOT: Wrong... N is a letter, that is not a number!
P1BOT: Turn 9, what is your guess?

Human: Z

P1BOT: Wrong... Z is a letter, that is not a number!
P1BOT: Turn 10, what is your guess?

Human:
```

When the user input is not a letter or a number, the program must respond as follows:



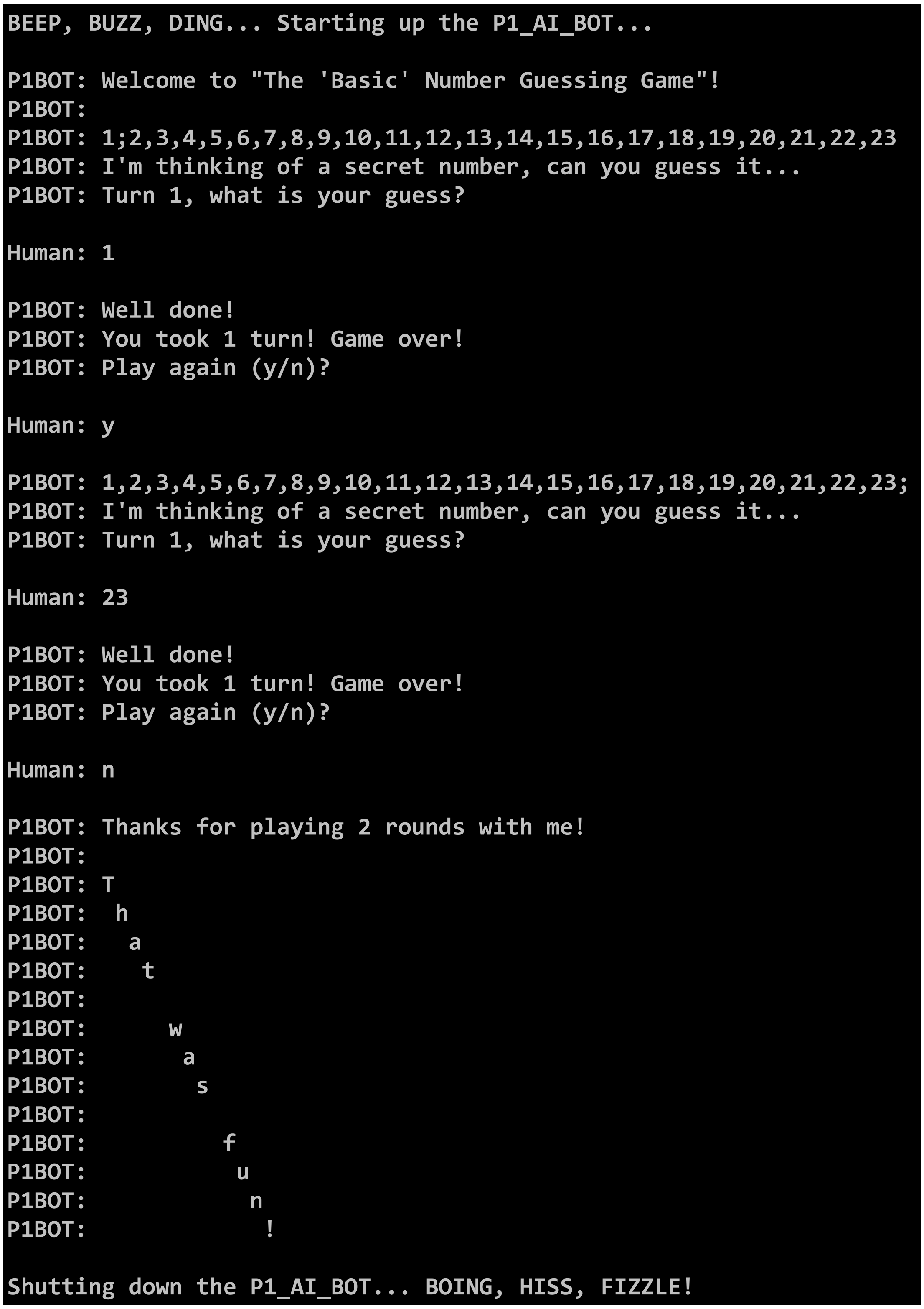
```
P1BOT: Turn 10, what is your guess?

Human: %

P1BOT: Wrong... wow, % is not a number!
P1BOT: Turn 11, what is your guess?

Human: *

P1BOT: Wrong... wow, * is not a number!
P1BOT: Turn 12, what is your guess?

Human: [

P1BOT: Wrong... wow, [ is not a number!
P1BOT: Turn 13, what is your guess?

Human:
```

An example of the completed program, with user input:



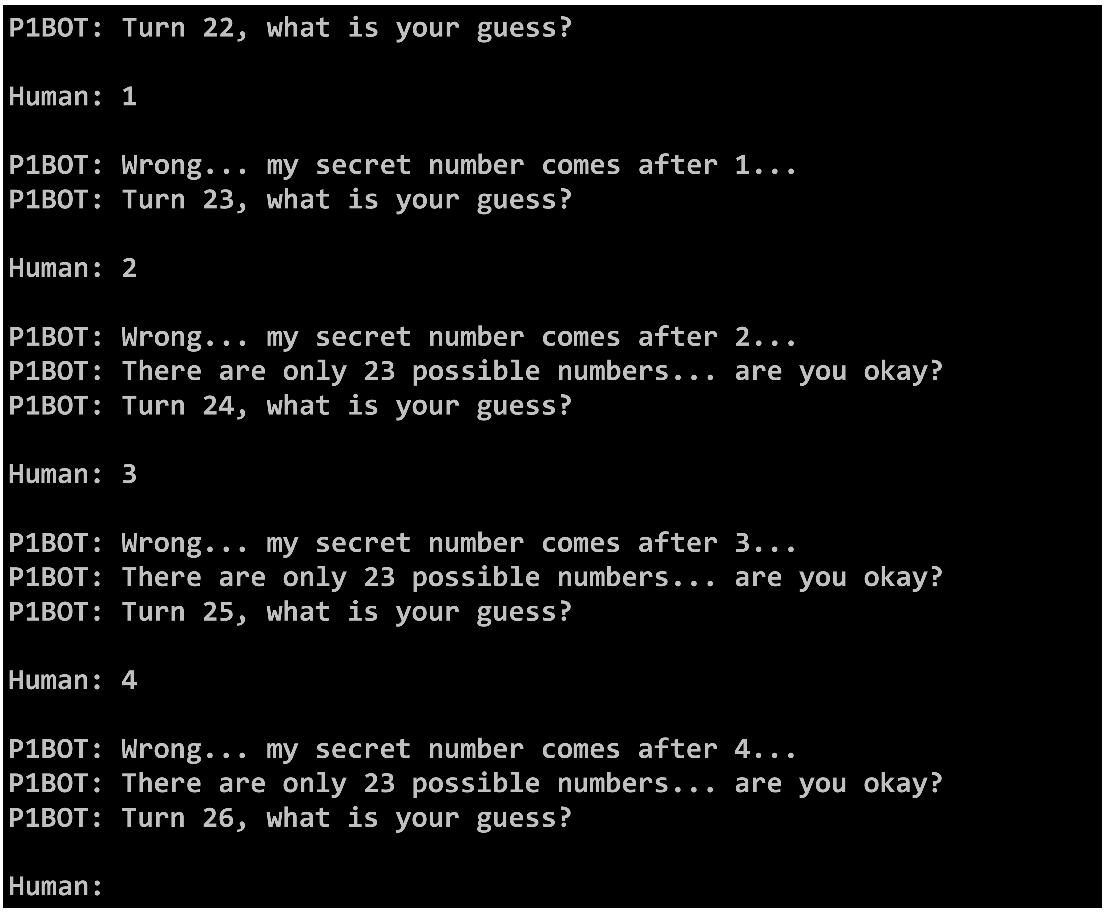
```
BEEP, BUZZ, DING... Starting up the P1_AI_BOT...

P1BOT: Welcome to "The 'Basic' Number Guessing Game"!
P1BOT:
P1BOT: 1,2,3,4,5,6,7,8,9,10,11,12,13;14,15,16,17,18,19,20,21,22,23
P1BOT: I'm thinking of a secret number, can you guess it...
P1BOT: Turn 1, what is your guess?

Human: 9

P1BOT: Wrong... my secret number comes after 9...
P1BOT: Turn 2, what is your guess?

Human: 22

P1BOT: Wrong... my secret number comes before 22...
P1BOT: Turn 3, what is your guess?

Human: 18

P1BOT: Wrong... my secret number comes before 18...
P1BOT: Turn 3, what is your guess?

Human: 13

P1BOT: Well done!
P1BOT: You took 4 turns! Game over!
P1BOT: Play again (y/n)?

Human: n

P1BOT: Thanks for playing 1 round with me!
P1BOT:
P1BOT: T
P1BOT:  h
P1BOT:   a
P1BOT:    t
P1BOT:
P1BOT:      w
P1BOT:       a
P1BOT:        s
P1BOT:
P1BOT:          f
P1BOT:           u
P1BOT:            n
P1BOT:             !

Shutting down the P1_AI_BOT... BOING, HISS, FIZZLE!
```

Another example of the completed program, with user input:



```
BEEP, BUZZ, DING... Starting up the P1_AI_BOT...

P1BOT: Welcome to "The 'Basic' Number Guessing Game"!
P1BOT:
P1BOT: 1;2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23
P1BOT: I'm thinking of a secret number, can you guess it...
P1BOT: Turn 1, what is your guess?

Human: 1

P1BOT: Well done!
P1BOT: You took 1 turn! Game over!
P1BOT: Play again (y/n)?

Human: y

P1BOT: 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23;
P1BOT: I'm thinking of a secret number, can you guess it...
P1BOT: Turn 1, what is your guess?

Human: 23

P1BOT: Well done!
P1BOT: You took 1 turn! Game over!
P1BOT: Play again (y/n)?

Human: n

P1BOT: Thanks for playing 2 rounds with me!
P1BOT:
P1BOT: T
P1BOT:  h
P1BOT:   a
P1BOT:    t
P1BOT:
P1BOT:      w
P1BOT:       a
P1BOT:        s
P1BOT:
P1BOT:          f
P1BOT:           u
P1BOT:            n
P1BOT:             !

Shutting down the P1_AI_BOT... BOING, HISS, FIZZLE!
```

One final requirement is that the simple AI must politely insult the player if they have taken more than 22 guesses at the secret number as follows:



```
P1BOT: Turn 22, what is your guess?

Human: 1

P1BOT: Wrong... my secret number comes after 1...
P1BOT: Turn 23, what is your guess?

Human: 2

P1BOT: Wrong... my secret number comes after 2...
P1BOT: There are only 23 possible numbers... are you okay?
P1BOT: Turn 24, what is your guess?

Human: 3

P1BOT: Wrong... my secret number comes after 3...
P1BOT: There are only 23 possible numbers... are you okay?
P1BOT: Turn 25, what is your guess?

Human: 4

P1BOT: Wrong... my secret number comes after 4...
P1BOT: There are only 23 possible numbers... are you okay?
P1BOT: Turn 26, what is your guess?

Human:
```

You must choose a **different polite insult** to occur after 22 guesses – do not use the example: "There are only 23 possible numbers... are you okay?" – choose your own unique message!

Based upon the program requirements set out above, start by creating a pseudo code design.

Declare and define at least five functions in addition to the **`main`** function. Do not use global variables or **`goto`**!

Once you have your pseudo code design, implement your design using C. Finally, robustly test your implementation to ensure it matches the requirements.

Follow good programming standards for code layout whitespace, naming and commenting. Ensure your C source code can successfully compile. Test your program with a variety of inputs and ensure the resulting program output is as described above.