

Laboratory 3/4 Notes

Hangman

The Game of Hangman

Welcome to the CS240 version of Hangman! Please enter the time (for example, if it is 8:30:32, enter 83032):

The word you are trying to guess is:

Guess a letter: **p**

Correct! The word is:

-pp--

Guess a letter: **x**

Incorrect!

—|

-pp--

Guess a letter: **z**

Incorrect!

—|
o

-pp--

Guess a letter: **y**

Incorrect!

—|
o
\
/

-pp--

Guess a letter: **k**

Incorrect!

```
—|  
  |  
  o  
  \|
```

-pp--

Guess a letter: **b**

Incorrect!

```
—|  
  |  
  o  
  \|/
```

-pp--

Guess a letter: **o**

Incorrect!

```
—|  
  |  
  o  
  \|/  
  /
```

-pp--

Guess a letter: **i**

Incorrect!

```
—|  
  |  
  o  
  \|/  
  / \
```

You're hung! The correct word is: apple

Would you like to play again? (y or n): y

The word you are trying to guess is:

Below is the general description of the steps you will need to follow to implement the game:

1. Generate a random value, and convert it to an index into the table of possible game words.
2. Get the new game word from the table, using the index.
3. Greet the user and show to them a word consisting of a number of blank spaces corresponding to the length of the game word.
4. Prompt them to guess a letter.
5. Get their guess and compare the letter with the game word.

If there is a match:

Display the matched letters (you should convert a guess entered in upper-case to a lower-case ascii letter before doing the comparison, since the table of game words are all lower-case)

If they have guessed all the letters:

Congratulate them, and ask them if they want to play again (go to step 7).

Else if they have not yet guessed all the letters:

Ask them for the next guess (back to step 4)

Else if there is not a match:

Display the hangman, one piece for each incorrect guess.

If the number of incorrect guesses is > 6 :

Send a 'you lose' message and ask them if they want to play again (go to step 7).

Else if the number of incorrect guesses is < 6 :

Ask them for the next guess (back to step 4).

6. When the game is over, ask the user if they want to play again. If so, go back to step 1. Otherwise, terminate the program.

In Lab today, you will program some of the tasks required to implement the Hangman game.

Main Program

In your main program, you will prompt the user for an initial seed value (this must be different each time the program is started, or the same sequence of words will be generated), and store the seed value in memory. You should only have to enter the seed value *once* (not for every new word you play).

The main program will then contain a loop that uses the seed value to generate an index to choose a game word. After the game is over, the program should ask the player if they want another game ('y' or 'n'). If 'y', repeat the loop. If 'n', thank the player and exit.

Generate Index into Array

The table of game words is an array of variable length strings. To get a game word from the table, you must generate an index 0 – 116, and access the game word associated with the index.

The following procedure **rand** uses a seed value to generate an integer in the range 0 – 116:

```
rand:
    lw $s0,seed      #load seed into $s0
    li $t0, 32749    #initialize large prime constants
    li $t1,32497

    li $t2,117      #initialize range of numbers to be generated

    multu $s0,$t0   #multiply seed by first large prime constant

    addu $t1,$s0,$t1 #add second constant to result and take
    remu $t1,$t1,$t2 #remainder after dividing by 117 to get index

    move $v0,$t1

    sw $v0,seed     #store new seed back into memory

    jr $ra
```

Upper to Lower case conversion

It turns out that upper and lower case letters are always 32 apart in the ascii table. So, you can test if a guessed letter has an ascii value lower than 'a' (meaning an uppercase letter was entered); if it does, you can simply add 32 to convert it to lowercase.

Guessing Letters

You should define a variable which will hold the string which represents the unguessed word (labeled *guessname*, for example).

At the start of each game, initialize *guessname* to dashes.

When you guess a letter that matches, replace the corresponding '-' in *guessname* with the correct letter. You can then simply output the *guessname* string to show the combination of dashes and letters.

Displaying the Hangman

You must keep a count of the number of wrong guesses. When a wrong guess is made, output a hangman figure corresponding to the number of errors made.

The complete hangman figure can be represented by a string **gallows**, whose data declaration is given below:

```
gallows: .asciiz "\n__\n |\n O\n \|/\n /\n"
```

You should not have to define separate strings for each version of the gallows. Instead, print out only the portion of the string corresponding to the number of errors.

Winning the Game

There are a number of ways to determine whether the player has won the game. It is up to you to come up with a strategy!