Project 1
(Hangman)

Taylor Marquez
CSC-11 48982
November 8, 2015

Introduction:

Title: Hangman!

Description: The game is a very simple variation of hangman. The player guesses letters to fill in the blanks of an unknown, randomly selected word. Six incorrect guesses make the player lose the game. In this version of the game there are only five different words, each six letters long. They are also all pokemon, for lack of a more proper theme.

Summary:

Total lines of code: 1013
Unique lines of code: 347
Repeated lines for words 2-5: 666
Variables: 12

The hardest part of putting together the project was coming up with the idea to work with, since the CSC5 class I took did not require a project of this sort. Once I had an idea to work with, it took roughly 10-12 hours over 5 days from the first line of my C prototype to the finished project.

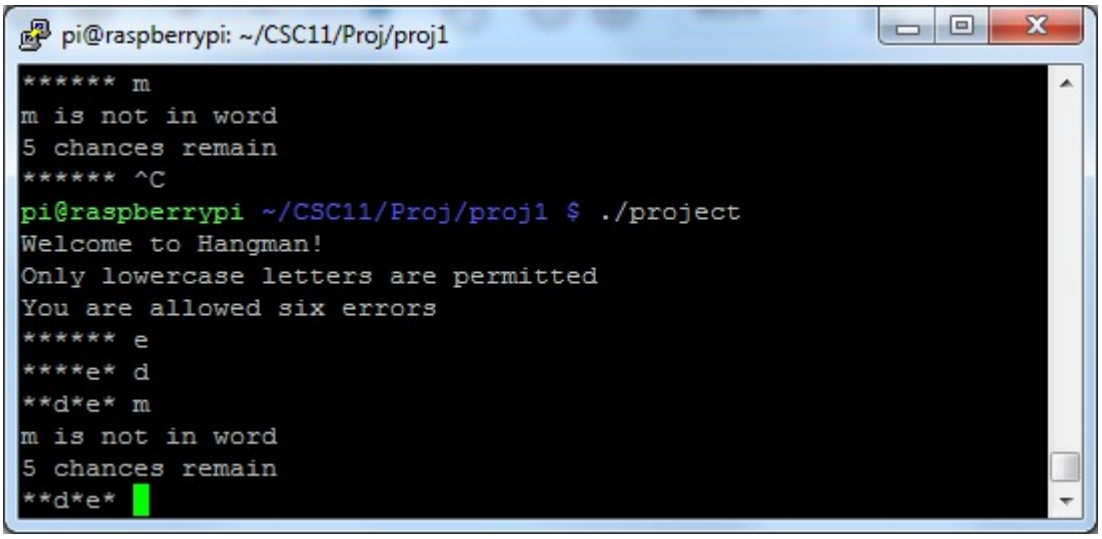
I do not feel that the project was particularly difficult to put together, since it consisted primarily of repurposing concepts from previous work, so almost all of my issues during development were merely careless mistakes that were simple to solve. I am quite satisfied with the final product. It covers what I was required to learn in the class up to this point. I believe it provides a very solid base to build from once we the rest of the material.

Description:

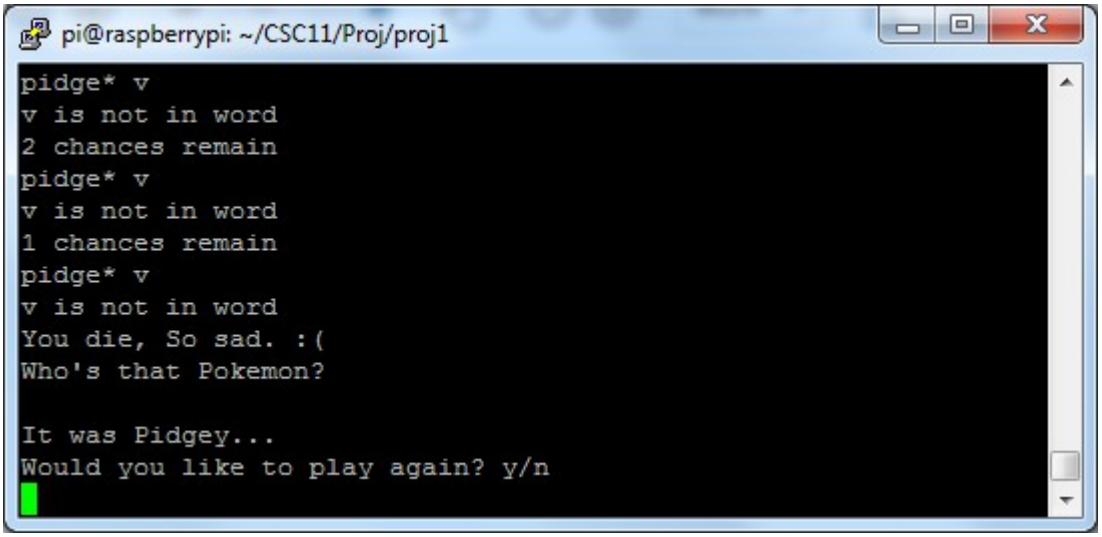
The main thing I did to complete the program was use if statements within a loop to determine how well the user input matches the hidden word.

Sample Input/Output:

When the game begins, the player is given a series of messages detailing the basic rules of the games. They are then required to input a lowercase letter. If an incorrect letter is guessed then the player is informed of their error, and how many more incorrect guesses they can make

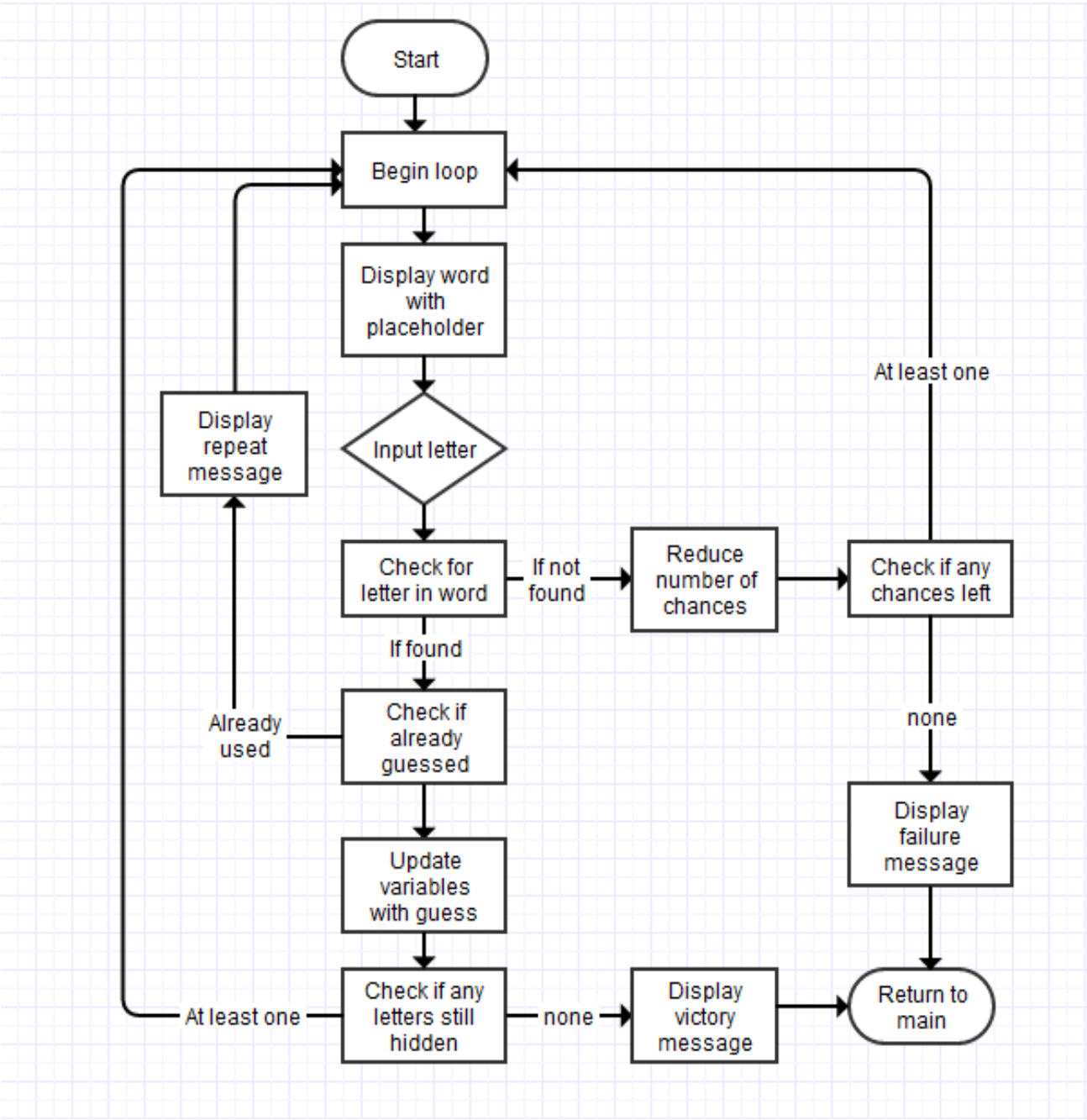


This continues until the player either guesses all the letters of the word, or uses up all their chances. They are then given an appropriate message, followed by a prompt to play again



Flowchart:

Generalized version of word0-4 flowchart



Psuedo-Code:

```
Initialize

Set R0 as random number between 0 and 4
Go to matching function

inside Function:
    Do
        Display variables
        Get input letter
        if input letter equals variable letter
            goto letter
        else
            goto letternotfound

    letter:
        if input letter equals register
            goto letterused
        register set to input letter
        unsolved letters minus 1
        goto checkifsolved

    letternotfound:
        chances minus 1
        if chances = 0
            goto lose
            endloop
        else
            loop again

    letterused:
        loop again

    checkifsolved:
        if unsolved letters = 0
            goto win
            endloop
        else
            loop
    while {chances >0 && unsolved letter >0}

    win:
        display win message
        end
```

lose:

display lose message
end

Variables:

main:

R1 – int
inReplay - char

word0-4:

inLetter – char
R4 - int
R5 – int
R6 – char
R7 – char
R8 – char
R9 – char
R10 – char
R11 – char

Language Constructs:

Chapter 3:
MOV intsruction - present in all files
source file – all files

Chapter 5:
link register – main, exit, word*, finish
BEQ instruction – main.s, word*.s

Chapter 6:
Arithetic Instructions – divmod, checkUnsolved, notFound

Chapter 7:
string – main.s, word*.s
LDR instruction – main.s, word*.s

Chapter 9:
conditional branching – main.s, word*.s

Chapter 10:
loop - “randomize” and word*.s “loop:”

Chapter 11:
ASR instruction - “randomize”

Others:
rand/srand – “randomize”
scanf – main, “replay,” “loop”
printf – main, word*
switch - “randomize”
if/else - “loop”
if - “letter*” “replay” “notfound” “checkUnsolved”

References:

1. textbook
2. ml1150258/LehrMark_CSC11_48982 (divmod)
3. http://www.asciitable.com/

Program:

```
 .data

.balign 4
scanpattern: .asciz " %c"

.balign 4
outstart1: .asciz "Welcome to Hangman!\n"

.balign 4
outstart2: .asciz "Only lowercase letters are permitted\nYou are allowed six
errors\n"

.balign 4
outReplay: .asciz "Would you like to play again? y/n\n"

.balign 4
inReplay: .word 0

.balign 4
rdnWord: .word 0

.text
```

```
        .global main

main:
        PUSH {r4,lr}

        LDR R0, =outstart1
        BL printf

        LDR R0, =outstart2
        BL printf

randomize:
        /*Set random value*/
        MOV R0, #0
        BL time
        BL srand
        BL rand
        MOV R1, R0, ASR #1
        MOV R2, #5
        BL divMod

        CMP R1, #0      @branch if random number was 0
        BEQ wrd0

        CMP R1, #1      @branch if random number was 1
        BEQ wrd1

        CMP R1, #2      @branch if random number was 2
        BEQ wrd2

        CMP R1, #3      @branch if random number was 3
        BEQ wrd3

        CMP R1, #4      @branch if random number was 4
        BEQ wrd4

wrd0:
        BL word0
        B replay

wrd1:
        BL word1
        B replay

wrd2:
```

```
        BL word2
        B replay

wrd3:
        BL word3
        B replay

wrd4:
        BL word4
        B replay

replay:
        LDR R0, =outReplay
        BL printf

        LDR R0, =scanpattern
        LDR R1, =inReplay
        BL scanf
        LDR R1, inReplayAddr
        LDR R1, [R1]

        CMP R1, #121
        BEQ randomize

        CMP R1, #110
        BEQ exit

exit:
        POP {r4,lr} @leave main
        BX lr

inReplayAddr: .word inReplay

/*external functions*/
.global scanf
.global printf
.global time
.global srand
.global rand

.data

.balign 4
scanPattern: .asciz " %c"

.balign 4
```

```
outLetter: .asciz "%c%c%c"

.balign 4
outLetter2: .asciz "%c%c%c "

.balign 4
inLetter: .word 0

.balign 4
outNotFound: .asciz "%c is not in word\n"

.balign 4
outChances: .asciz "%d chances remain\n"

.balign 4
outUsed: .asciz "%c has already been used\n"

.balign 4
outFailure: .asciz "You die, So sad. :(\nWho's that Pokemon?\n\nIt was
Pidgey...\n"

.balign 4
outSuccess: .asciz "You're Winner!\nWho's that Pokemon?\n\nIts Pidgey!!\n"

.text

        .global word0

word0:
        PUSH {lr}

        MOV R4, #6              @remaining chances
        MOV R5, #6              @unsolved letters
        MOV R6, #42             @'*' as placeholer for unsolved letters
        MOV R7, #42
        MOV R8, #42
        MOV R9, #42
        MOV R10, #42
        MOV R11, #42

loop:
        /*Display in sets of three*/
        LDR R0, =outLetter
        MOV R1, R6              @first letter
        MOV R2, R7              @second letter
        MOV R3, R8              @third letter
```

```
        BL printf

        LDR R0, =outLetter2
        MOV R1, R9              @fourth letter
        MOV R2, R10             @fifth letter
        MOV R3, R11             @sixth letter
        BL printf

        LDR R0, =scanPattern
        LDR R1, =inLetter
        BL scanf
        LDR R1, inLetterAddr
        LDR R1, [R1]

        CMP R1, #112            @check if inLetter = 'p'
        BEQ letterp

        CMP R1, #105            @check if inLetter = 'i'
        BEQ letteri

        CMP R1, #100            @check if inLetter = 'd'
        BEQ letterd

        CMP R1, #103            @check if inLetter = 'g'
        BEQ letterg

        CMP R1, #101            @check if inLetter = 'e'
        BEQ lettere

        CMP R1, #121            @check if inLetter = 'y'
        BEQ lettery

        B notFound          @branch if none of the above

letterp:
        CMP R6, R1          @check if used already
        BEQ used
        MOV R6, R1
        SUB R5, R5, #1
        B checkUnsolved

letteri:
        CMP R7, R1          @check if used already
        BEQ used
        MOV R7, R1
        SUB R5, R5, #1
```

```
        B checkUnsolved

letterd:
        CMP R8, R1          @check if used already
        BEQ used
        MOV R8, R1
        SUB R5, R5, #1
        B checkUnsolved

letterg:
        CMP R9, R1          @check if used already
        BEQ used
        MOV R9, R1
        SUB R5, R5, #1
        B checkUnsolved

lettere:
        CMP R10, R1                 @check if used already
        BEQ used
        MOV R10, R1
        SUB R5, R5, #1
        B checkUnsolved

lettery:
        CMP R11, R1                 @check if used already
        BEQ used
        MOV R11, R1
        SUB R5, R5, #1
        B checkUnsolved

notFound:
        /*Display message for incorrect guesses*/
        LDR R0, =outNotFound
        BL printf

        SUB R4, R4, #1              @R4--
        CMP R4, #0                  @check if any chances remain
        BLE failure

        /*Display message for remaining chances*/
        LDR R0, =outChances
        MOV R1, R4
        BL printf

        B loop                      @return to loop if any chances remain
```

```
checkUnsolved:
        CMP R5, #0              @check if all letters solved
        BLE success
        B loop                  @return to loop if not

used:
        /*Display message for repeat guesses*/
        LDR R0, =outUsed
        BL printf
        B loop

failure:
        /*Display message for failed game*/
        LDR R0, =outFailure
        BL printf

        B finish

success:
        /*Display message for successful game*/
        LDR R0, =outSuccess
        BL printf

        B finish

finish:
        /*return to main*/
        POP {lr}
        BX lr

inLetterAddr: .word inLetter

.data

.balign 4
scanPattern: .asciz " %c"

.balign 4
outLetter: .asciz "%c%c%c"

.balign 4
outLetter2: .asciz "%c%c%c "

.balign 4
inLetter: .word 0
```

```
.balign 4
outNotFound: .asciz "%c is not in word\n"

.balign 4
outChances: .asciz "%d chances remain\n"

.balign 4
outUsed: .asciz "%c has already been used\n"

.balign 4
outFailure: .asciz "You die, So sad. :(\nWho's that Pokemon?\n\nIt was
Mewtwo...\n"

.balign 4
outSuccess: .asciz "You're Winner!\nWho's that Pokemon?\n\nIts Mewtwo!!\n"

.text

        .global word1

word1:
        PUSH {lr}

        MOV R4, #6              @remaining chances
        MOV R5, #6              @unsolved letters
        MOV R6, #42             @'*' as placeholer for unsolved letters
        MOV R7, #42
        MOV R8, #42
        MOV R9, #42
        MOV R10, #42
        MOV R11, #42

loop:
        LDR R0, =outLetter
        MOV R1, R6              @first letter
        MOV R2, R7              @second letter
        MOV R3, R8              @third letter
        bl printf

        LDR R0, =outLetter2
        MOV R1, R9              @fourth letter
        MOV R2, R10             @five letter
        MOV R3, R11             @sixth letter
        BL printf

        LDR R0, =scanPattern
```

```
        LDR R1, =inLetter
        BL scanf
        LDR R1, inLetterAddr
        LDR R1, [R1]

        CMP R1, #109            @check if inLetter = 'm'
        BEQ letterm

        CMP R1, #101            @check if inLetter = 'e'
        BEQ lettere

        CMP R1, #119            @check if inLetter = 'w'
        BEQ letterw

        CMP R1, #116            @check if inLetter = 't'
        BEQ lettert

        CMP R1, #111            @check if inLetter = 'o'
        BEQ lettero

        B notFound          @branch if none of the above

letterm:
        CMP R6, R1          @check if used already
        BEQ used
        MOV R6, R1
        SUB R5, R5, #1
        B checkUnsolved

lettere:
        CMP R7, R1          @check if used already
        BEQ used
        MOV R7, R1
        SUB R5, R5, #1
        B checkUnsolved

letterw:
        CMP R8, R1          @check if used already
        BEQ used
        MOV R8, R1
        MOV R10, R1
        SUB R5, R5, #2
        B checkUnsolved

lettert:
        CMP R9, R1          @check if used already
```

```
        BEQ used
        MOV R9, R1
        SUB R5, R5, #1
        B checkUnsolved

lettero:
        CMP R11, R1             @check if used already
        BEQ used
        MOV R11, R1
        SUB R5, R5, #1
        B checkUnsolved

notFound:
        /*Display message for incorrect guesses*/
        LDR R0, =outNotFound
        BL printf

        SUB R4, R4, #1          @R4--
        CMP R4, #0              @check if any chances remain
        BLE failure

        /*Display message for remaining chances*/
        LDR R0, =outChances
        MOV R1, R4
        BL printf

        B loop                  @return to loop if any chances remain

checkUnsolved:
        CMP R5, #0              @check if all letters solved
        BLE success
        B loop                  @return to loop if not

used:
        /*Display message for repeat guesses*/
        LDR R0, =outUsed
        BL printf
        B loop

failure:
        /*Display message for failed game*/
        LDR R0, =outFailure
        BL printf

        B finish
```

```
success:
        /*Display message for successful game*/
        LDR R0, =outSuccess
        BL printf

        B finish

finish:
        /*return to main*/
        POP {lr}
        BX lr

inLetterAddr: .word inLetter

.data

.balign 4
scanPattern: .asciz " %c"

.balign 4
outLetter: .asciz "%c%c%c"

.balign 4
outLetter2: .asciz "%c%c%c "

.balign 4
inLetter: .word 0

.balign 4
outNotFound: .asciz "%c is not in word\n"

.balign 4
outChances: .asciz "%d chances remain\n"

.balign 4
outUsed: .asciz "%c has already been used\n"

.balign 4
outFailure: .asciz "You die, So sad. :(\nWho's that Pokemon?\n\nIt was
Meowth...\n"

.balign 4
outSuccess: .asciz "You're Winner!\nWho's that Pokemon?\n\nIts Meowth!!\n"

.text
```

```
        .global word2

word2:
        PUSH {lr}

        MOV R4, #6              @remaining chances
        MOV R5, #6              @unsolved letters
        MOV R6, #42             @'*' as placeholer for unsolved letters
        MOV R7, #42
        MOV R8, #42
        MOV R9, #42
        MOV R10, #42
        MOV R11, #42

loop:
        LDR R0, =outLetter
        MOV R1, R6              @first letter
        MOV R2, R7              @second letter
        MOV R3, R8              @third letter
        bl printf

        LDR R0, =outLetter2
        MOV R1, R9              @fourth letter
        MOV R2, R10             @fifth letter
        MOV R3, R11             @sixth letter
        BL printf

        LDR R0, =scanPattern
        LDR R1, =inLetter
        BL scanf
        LDR R1, inLetterAddr
        LDR R1, [R1]

        CMP R1, #109            @check if inLetter = 'm'
        BEQ letterm

        CMP R1, #101            @check if inLetter = 'e'
        BEQ lettere

        CMP R1, #111            @check if inLetter = 'o'
        BEQ lettero

        CMP R1, #119            @check if inLetter = 'w'
        BEQ letterw

        CMP R1, #116            @check if inLetter = 't'
```

```
        BEQ lettert

        CMP R1, #104            @check if inLetter = 'h'
        BEQ letterh

        B notFound          @branch if none of the above

letterm:
        CMP R6, R1          @check if used already
        BEQ used
        MOV R6, R1
        SUB R5, R5, #1
        B checkUnsolved

lettere:
        CMP R7, R1          @check if used already
        BEQ used
        MOV R7, R1
        SUB R5, R5, #1
        B checkUnsolved

lettero:
        CMP R8, R1          @check if used already
        BEQ used
        MOV R8, R1
        SUB R5, R5, #1
        B checkUnsolved

letterw:
        CMP R9, R1          @check if used already
        BEQ used
        MOV R9, R1
        SUB R5, R5, #1
        B checkUnsolved

lettert:
        CMP R10, R1             @check if used already
        BEQ used
        MOV R10, R1
        SUB R5, R5, #1
        B checkUnsolved

letterh:
        CMP R11, R1             @check if used already
        BEQ used
        MOV R11, R1
```

```
        SUB R5, R5, #1
        B checkUnsolved

notFound:
        /*Display message for incorrect guesses*/
        LDR R0, =outNotFound
        BL printf

        SUB R4, R4, #1          @R4--
        CMP R4, #0              @check if any chances remain
        BLE failure

        /*Display message for remaining chances*/
        LDR R0, =outChances
        MOV R1, R4
        BL printf

        B loop                  @return to loop if any chances remain

checkUnsolved:
        CMP R5, #0              @check if all letters solved
        BLE success
        B loop                  @return to loop if not

used:
        /*Display message for repeat guesses*/
        LDR R0, =outUsed
        BL printf
        B loop

failure:
        /*Display message for failed game*/
        LDR R0, =outFailure
        BL printf

        B finish

success:
        /*Display message for successful game*/
        LDR R0, =outSuccess
        BL printf

        B finish

finish:
        /*return to main*/
```

```
        POP {lr}
        BX lr

inLetterAddr: .word inLetter

.data

.balign 4
scanPattern: .asciz " %c"

.balign 4
outLetter: .asciz "%c%c%c"

.balign 4
outLetter2: .asciz "%c%c%c "

.balign 4
inLetter: .word 0

.balign 4
outNotFound: .asciz "%c is not in word\n"

.balign 4
outChances: .asciz "%d chances remain\n"

.balign 4
outUsed: .asciz "%c has already been used\n"

.balign 4
outFailure: .asciz "You die, So sad. :(\nWho's that Pokemon?\n\nIt was
Gengar...\n"

.balign 4
outSuccess: .asciz "You're Winner!\nWho's that Pokemon?\n\nIts Gengar!!\n"


.text

        .global word3

word3:
        PUSH {lr}

        MOV R4, #6              @remaining chances
        MOV R5, #6              @unsolved letters
        MOV R6, #42             @'*' as placeholer for unsolved letters
```

```
        MOV R7, #42
        MOV R8, #42
        MOV R9, #42
        MOV R10, #42
        MOV R11, #42

loop:
        LDR R0, =outLetter
        MOV R1, R6              @first letter
        MOV R2, R7              @second letter
        MOV R3, R8              @third letter
        bl printf

        LDR R0, =outLetter2
        MOV R1, R9              @fourth letter
        MOV R2, R10             @fifth letter
        MOV R3, R11             @sixth letter
        BL printf

        LDR R0, =scanPattern
        LDR R1, =inLetter
        BL scanf
        LDR R1, inLetterAddr
        LDR R1, [R1]

        CMP R1, #103            @check if inLetter = 'g'
        BEQ letterg

        CMP R1, #101            @check if inLetter = 'e'
        BEQ lettere

        CMP R1, #110            @check if inLetter = 'n'
        BEQ lettern

        CMP R1, #97             @check if inLetter = 'a'
        BEQ lettera

        CMP R1, #114            @check if inLetter = 'r'
        BEQ letterr

        B notFound        @branch if none of the above

letterg:
        CMP R6, R1        @check if used already
        BEQ used
        MOV R6, R1
```

```
        MOV R9, R1
        SUB R5, R5, #2
        B checkUnsolved

lettere:
        CMP R7, R1          @check if used already
        BEQ used
        MOV R7, R1
        SUB R5, R5, #1
        B checkUnsolved

lettern:
        CMP R8, R1          @check if used already
        BEQ used
        MOV R8, R1
        SUB R5, R5, #1
        B checkUnsolved

lettera:
        CMP R10, R1                 @check if used already
        BEQ used
        MOV R10, R1
        SUB R5, R5, #1
        B checkUnsolved

letterr:
        CMP R11, R1                 @check if used already
        BEQ used
        MOV R11, R1
        SUB R5, R5, #1
        B checkUnsolved

notFound:
        /*Display message for incorrect guesses*/
        LDR R0, =outNotFound
        BL printf

        SUB R4, R4, #1              @R4--
        CMP R4, #0                  @check if any chances remain
        BLE failure

        /*Display message for remaining chances*/
        LDR R0, =outChances
        MOV R1, R4
        BL printf
```

```
        B loop                  @return to loop if any chances remain

checkUnsolved:
        CMP R5, #0              @check if all letters solved
        BLE success
        B loop                  @return to loop if not

used:
        /*Display message for repeat guesses*/
        LDR R0, =outUsed
        BL printf
        B loop

failure:
        /*Display message for failed game*/
        LDR R0, =outFailure
        BL printf

        B finish

success:
        /*Display message for successful game*/
        LDR R0, =outSuccess
        BL printf

        B finish

finish:
        /*return to main*/
        POP {lr}
        BX lr

inLetterAddr: .word inLetter

.data

.balign 4
scanPattern: .asciz " %c"

.balign 4
outLetter: .asciz "%c%c%c"

.balign 4
outLetter2: .asciz "%c%c%c "

.balign 4
```

```
inLetter: .word 0

.balign 4
outNotFound: .asciz "%c is not in word\n"

.balign 4
outChances: .asciz "%d chances remain\n"

.balign 4
outUsed: .asciz "%c has already been used\n"

.balign 4
outFailure: .asciz "You die, So sad. :(\nWho's that Pokemon?\n\nIt was
Espurr...\n"

.balign 4
outSuccess: .asciz "You're Winner!\nWho's that Pokemon?\n\nIts Espurr!!\n"

.text

        .global word4

word4:
        PUSH {lr}

        MOV R4, #6              @remaining chances
        MOV R5, #6              @unsolved letters
        MOV R6, #42             @'*' as placeholer for unsolved letters
        MOV R7, #42
        MOV R8, #42
        MOV R9, #42
        MOV R10, #42
        MOV R11, #42

loop:
        LDR R0, =outLetter
        MOV R1, R6              @first letter
        MOV R2, R7              @second letter
        MOV R3, R8              @third letter
        bl printf

        LDR R0, =outLetter2
        MOV R1, R9              @fourth letter
        MOV R2, R10             @fifth letter
        MOV R3, R11             @sixth letter
        BL printf
```

```
        LDR R0, =scanPattern
        LDR R1, =inLetter
        BL scanf
        LDR R1, inLetterAddr
        LDR R1, [R1]

        CMP R1, #101            @check if inLetter = 'e'
        BEQ lettere

        CMP R1, #115            @check if inLetter = 's'
        BEQ letters

        CMP R1, #112            @check if inLetter = 'p'
        BEQ letterp

        CMP R1, #117            @check if inLetter = 'u'
        BEQ letteru

        CMP R1, #114            @check if inLetter = 'r'
        BEQ letterr

        B notFound          @branch if none of the above

lettere:
        CMP R6, R1          @check if used already
        BEQ used
        MOV R6, R1
        SUB R5, R5, #1
        B checkUnsolved

letters:
        CMP R7, R1          @check if used already
        BEQ used
        MOV R7, R1
        SUB R5, R5, #1
        B checkUnsolved

letterp:
        CMP R8, R1          @check if used already
        BEQ used
        MOV R8, R1
        SUB R5, R5, #1
        B checkUnsolved

letteru:
```

```
        CMP R9, R1          @check if used already
        BEQ used
        MOV R9, R1
        SUB R5, R5, #1
        B checkUnsolved

letterr:
        CMP R10, R1                 @check if used already
        BEQ used
        MOV R10, R1
        MOV R11, R1
        SUB R5, R5, #2
        B checkUnsolved

notFound:
        /*Display message for incorrect guesses*/
        LDR R0, =outNotFound
        BL printf

        SUB R4, R4, #1              @R4--
        CMP R4, #0                  @check if any chances remain
        BLE failure

        /*Display message for remaining chances*/
        LDR R0, =outChances
        MOV R1, R4
        BL printf

        B loop                      @return to loop if any chances remain

checkUnsolved:
        CMP R5, #0                  @check if all letters solved
        BLE success
        B loop                      @return to loop if not

used:
        /*Display message for repeat guesses*/
        LDR R0, =outUsed
        BL printf
        B loop

failure:
        /*Display message for failed game*/
        LDR R0, =outFailure
        BL printf
```

```
        B finish

success:
        /*Display message for successful game*/
        LDR R0, =outSuccess
        BL printf

        B finish

finish:
        /*return to main*/
        POP {lr}
        BX lr

inLetterAddr: .word inLetter

/*
        Functions
                scaleRight
                addSub
                scaleLeft
                divMod
*/

.text

/*void scaleRight(int &r1,int &r3,int &r2) */
.globl scaleRight
scaleRight:
        push {lr}             /* Push lr onto the stack */
        doWhile_r1_lt_r2:     /* Shift right until just under the remainder */
                mov r3,r3,ASR #1; /* Division counter */
                mov r2,r2,ASR #1  /* Mod/Remainder subtraction */
        cmp r1,r2
        blt doWhile_r1_lt_r2
        pop {lr}              /* Pop lr from the stack */
    bx lr                /* Leave scaleRight */
/* end scaleRight */

/* void addSub(int &r3,int &r2,int &r0,int &r1) */
.globl addSub
addSub:
        push {lr}       /* Push lr onto the stack */
        doWhile_r3_ge_1:
                add r0,r0,r3
                sub r1,r1,r2
```

```
            bl scaleRight
        cmp r3,#1
        bge doWhile_r3_ge_1
    pop {lr}       /* Pop lr from the stack */
    bx lr          /* Leave addSub */
/* end addSub */

/* void scaleLeft(int &r1,int &r3,int &r2) */
.globl scaleLeft
scaleLeft:
        push {lr}             /* Push lr onto the stack */
        doWhile_r1_ge_r2:     /* Scale left till overshoot with remainder */
            mov r3,r3,LSL #1  /* scale factor */
            mov r2,r2,LSL #1  /* subtraction factor */
            cmp r1,r2
        bge doWhile_r1_ge_r2  /* End loop at overshoot */
        mov r3,r3,ASR #1      /* Scale factor back */
        mov r2,r2,ASR #1      /* Scale subtraction factor back */
        pop {lr}              /* Pop lr from the stack */
    bx lr                /* Leave addSub */
/* end scaleLeft */

/* void divMod(int &r2,int &r0,int &r1) */
.globl divMod
divMod:
        push {lr}       /* Push lr onto the stack */
        /* Determine the quotient and remainder */
        mov r0,#0
        mov r3,#1
        cmp r1,r2
        blt end
            bl scaleLeft
            bl addSub
        end:
        pop {lr}       /* Pop lr from the stack */
    bx lr          /* Leave addSub */
/* end divMod */
```