

Hanged Man Game

DP1-1 Eric, Rifle, Noel, Leo

May 22, 2020

Abstract In our paper, we designed a computer game called "Hanged Man". We utilized top-down programming methodology, dividing our design into several modules.

This paper includes the structure of our design, flowchart and pseudocode of each module, and c++ program implementation.

Contents

1	Introduction	3
2	Structure	4
3	Flowchart	5
4	Pseudocode	11
5	C++ implementation	16
6	Testing of implementation	30

1 Introduction

Hanged Man is a game. One player enter the hardness, word, and hint, another player try characters and if the character he / she input is in the word, all same characters in the word would appear; if the character is NOT in the word, a stroke is added to a hanged man. When all the characters are tried out, player 2 win; when the hanged man are completely drawn, player 2 die.

In our design, we set the relationship of hardness and uncomplete stroke as:

Hardness	Number of uncompleted strokes
SIMPLE	10
MIDDLE	9
HARD	8

We set the maximum number of characters of word is 20

We set the maximum number of characters of hint is 100

All the works of this project are uploaded to github repository <https://github.com/EricEricEricJin/Hanged-Man-Game.git> (<https://github.com/EricEricEricJin/Hanged-Man-Game.git>)

2 Structure

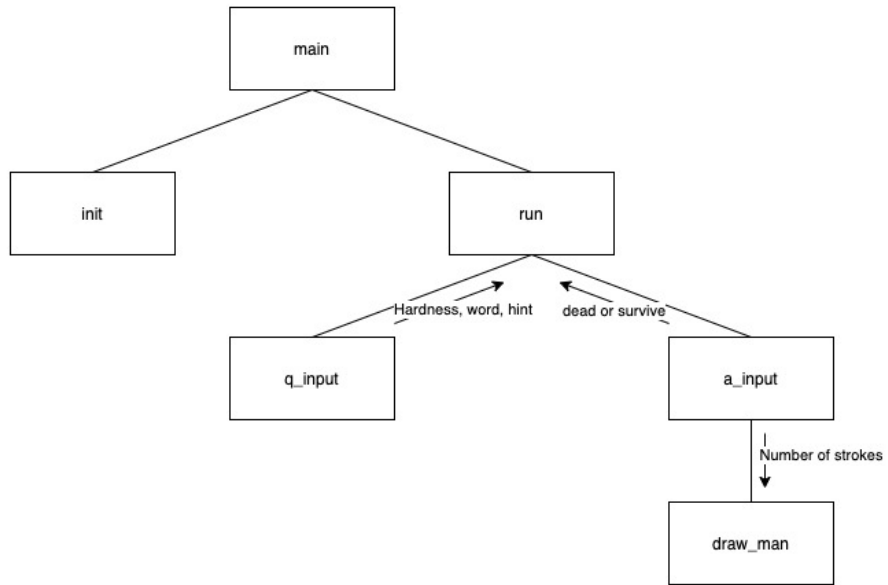


Figure 1: Structure chart

3 Flowchart

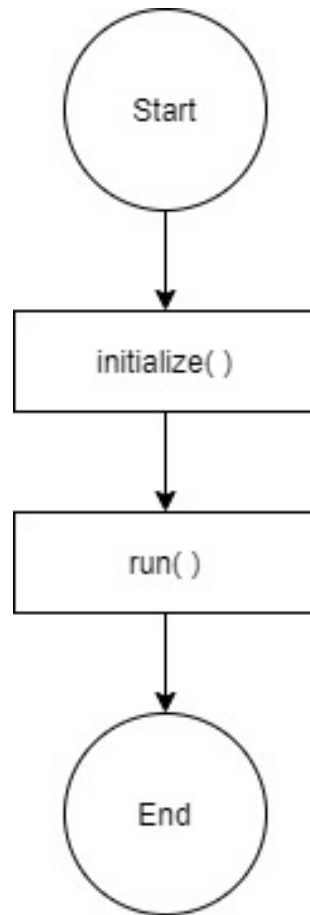


Figure 2: main

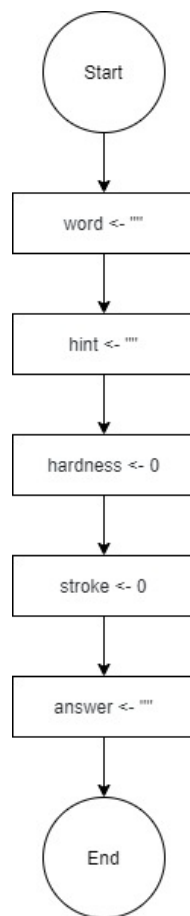


Figure 3: initialize

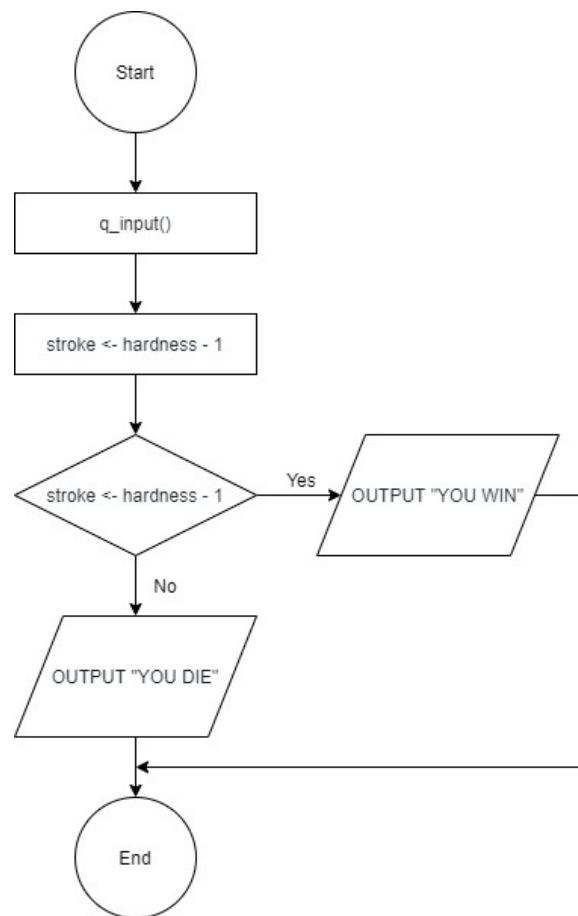


Figure 4: run

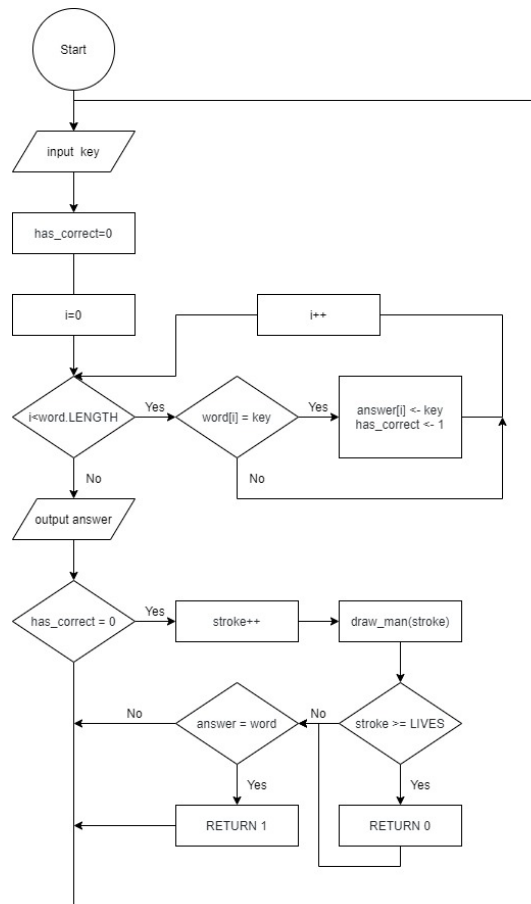


Figure 5: q_input

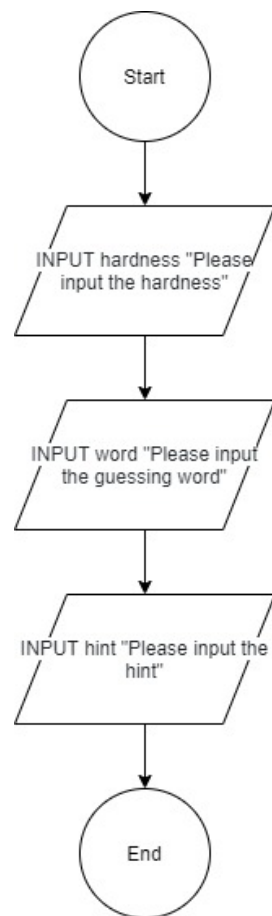


Figure 6: a.input

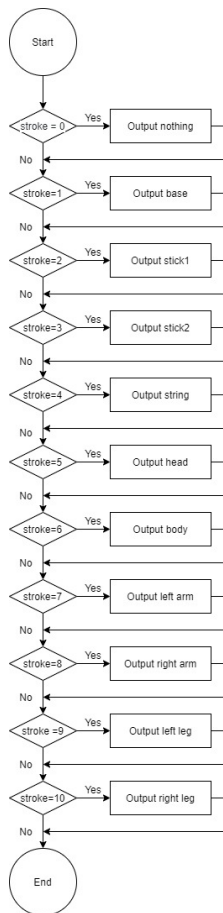


Figure 7: draw_man

4 Pseudocode

main

```
1 initialize ()
2 run ()
```

init

```
1 PROC initialize ()
2   DECLARE word: STRING
3   DECLARE hint: STRING
4   DECLARE hardness: INTEGER
5   DECLARE stroke: INTEGER
6   DECLARE answer: STRING
7
8   word <- ""
9   hint <- ""
10  hardness <- 0
11  stroke <- 0
12  answer <- ""
13 ENDPROC
```

run

```
1 PROC run ()
2   q_input ()
3   stroke <- hardness - 1
4   IF a_input () = 1 THEN
5     OUTPUT "YOU WIN"
6   ELSE
7     OUTPUT "YOU DIE"
8   ENDIF
9 ENDPROC
```

q_input

```
1 PROC q_input ()
2   INPUT hardness "Please input the hardness"
3   INPUT word "Please input the guessing word"
4   INPUT hint "Please input the hint"
5 ENDPROC
```

a_input

```
1 PROC a_input ()
2   DECLEAR key: CHARACTER
3   DECLEAR i: INTEGER
```

```

4  DECLARE has_correct: BOOLEAN
5  key <- ""
6
7  WHILE
8      INPUT key
9      has_correct <- 0
10     FOR i <- 0 TO word.LENGTH
11         IF word[i] = key THEN
12             answer[i] <- key
13             has_correct <- 1
14         ENDIF
15     ENDFOR
16     OUTPUT answer
17     IF has_correct = 0 THEN
18         stroke <- stroke + 1
19         draw_man(stroke)
20         IF stroke >= LIVES THEN
21             // loose
22             RETURN 0;
23         ENDIF
24     ELIF answer = word THEN
25         // win
26         RETURN 1;
27     ENDIF
28 ENDWHILE
29 ENDPROC

```

draw_man

```

1  PROC draw_man()
2      IF stroke == 0 THEN
3          OUTPUT
4              "
5
6
7
8
9
10
11
12              "
13
14     ELIF stroke == 1 THEN
15         OUTPUT
16             "
17

```

```

18
19
20
21
22
23
24         *****
25     "
26
27     ELIF stroke == 2 THEN
28         OUTPUT
29     "
30         *
31         *
32         *
33         *
34         *
35         *
36         *
37         *****
38     "
39
40     ELIF stroke == 3 THEN
41         OUTPUT
42     "
43         *****
44         *
45         *
46         *
47         *
48         *
49         *
50         *****
51     "
52
53     ELIF stroke == 4 THEN
54         OUTPUT
55     "
56         *****
57         *      *
58         *
59         *
60         *
61         *
62         *
63         *****

```

```

64      "
65
66      ELIF stroke == 5 THEN
67          OUTPUT
68          "
69              *****
70              *      *
71              *    * *
72              *      *
73              *
74              *
75              *
76              *****
77          "
78
79      ELIF stroke == 6 THEN
80          OUTPUT
81          "
82              *****
83              *      *
84              *    * *
85              *      *
86              *      *
87              *      *
88              *
89              *****
90          "
91      ELIF stroke == 7 THEN
92          OUTPUT
93          "
94              *****
95              *      *
96              *    * *
97              *      *
98              *    **
99              *      *
100             *
101             *****
102         "
103     ELIF stroke == 8 THEN
104         OUTPUT
105         "
106             *****
107             *      *
108             *    * *
109             *      *

```

```

110      *   ***
111      *     *
112      *
113      *
114      *
115      *
116      *
117      *
118      *
119      *
120      *
121      *
122      *
123      *
124      *
125      *
126      *
127      *
128      *
129      *
130      *
131      *
132      *
133      *
134      *
135      *
136      *
137      *
138      *
139      *
140      *
141      *
142      *

```

5 C++ implementation

hanged_man.h

```
1 #ifndef _H_M_
2 #define _H_M_
3
4 #define WINDOW_H 24
5 #define WINDOW_W 80
6
7 #define WORD_MAXLEN 20
8 #define HINT_MAXLEN 100
9
10 #define LIVES 10
11
12 #include <cstdio>
13 #include <cstdlib>
14 #include < curses.h>
15
16 struct string {
17     char* text;
18     int len;
19 };
20
21 class hangedMan {
22     private:
23         WINDOW* main_win;
24         WINDOW* man_win;
25         WINDOW* IO_win;
26
27         int q_input_stage;
28         int hardness;
29         string* hint;
30         string* word;
31         string* answer;
32
33         int stroke;
34
35
36         int restart;
37
38         void draw_man();
39         int q_input(int key_val);
40         int a_input(int key_val);
41         int strcmp(char* s1, char* s2, int len);
42         void die();
```



```

43     void win();
44     void del();
45     int rst_input(int key_val);
46     string* init_str(int len, char chr);
47     void free_str(string* str);
48
49     public:
50         hangedMan();
51         void run();
52
53 };
54 #endif

```

main.cpp

```

1  #include "hanged_man.h"
2
3  using namespace std;
4  int main() {
5      hangedMan H;
6      H.run();
7  }

```

init.cpp

```

1  /**
2   * Function: init
3   * Description: initialize as the class is instantiated
4   * Parameter: None
5   * Return: None
6   */
7
8  #include "hanged_man.h"
9
10 hangedMan::hangedMan() {
11     main_win = initscr();
12     man_win = subwin(main_win, LINES, int(COLS / 2), 0,
13                     int(COLS / 2));
14     IO_win = subwin(main_win, LINES, int(COLS / 2), 0, 0)
15     ;
16     // cbreak();
17     noecho();
18 }

```

run.cpp

```

1  /**

```

```

2  * Function: run
3  * Description: the main loop of the game
4  * Parameter: None
5  * Return: None
6  */
7
8  #include "hanged_man.h"
9
10 void hangedMan::run() {
11     word = init_str(WORD_MAXLEN, 0);
12     hint = init_str(HINT_MAXLEN, 0);
13     q_input_stage = 0;
14     hardness = 0;
15     stroke = 0;
16     restart = 0;
17
18     wclear(main_win);
19     wclear(man_win);
20     wclear(IO_win);
21
22     box(main_win, ACS_VLINE, ACS_HLINE);
23
24     int key_val = 0;
25     q_input_stage = 0;
26     while (1) {
27
28         if (q_input(key_val) == 1) {
29             break;
30         }
31         key_val = getch();
32     }
33
34     wclear(main_win);
35     box(main_win, ACS_VLINE, ACS_HLINE);
36     box(man_win, ACS_VLINE, ACS_HLINE);
37     box(IO_win, ACS_VLINE, ACS_HLINE);
38
39     stroke += hardness - 1;
40     answer = init_str(word -> len, 95);
41
42     wprintw(IO_win, answer -> text);
43     key_val = 0;
44
45     int brk = 0;
46     while (1) {
47         if (brk == 1) {

```

```

48         break;
49     }
50     switch (a_input(key_val)) {
51         case 1:
52             win();
53             brk = 1;
54             break;
55         case -1:
56             brk = 1;
57             die();
58             break;
59         default:
60             key_val = getch();
61             break;
62     }
63 }
64
65 // restart or quit
66 getch();
67 key_val = 0;
68 restart = 0;
69 while (1) {
70     if (rst_input(key_val) == 1) {
71         if (restart == 0) {
72             del();
73         } else if (restart == 1) {
74             del();
75             run();
76         }
77         break;
78     }
79     key_val = getch();
80 }
81 }

```

a_input.cpp

```

1  /**
2  * Function: a_input
3  * Description: the player input the answer
4  * Parameter: key_val
5  * Return: die(-1) or win(1) or unfinished(0)
6  */
7
8  #include "hanged_man.h"
9

```

```

10 int hangedMan::a_input(int key_val) {
11     if (key_val != 0) {
12         int hasCorrect = 0;
13         for (int i = 0; i < word -> len; i++) {
14             if (*(word -> text + i) == key_val) {
15                 hasCorrect = 1;
16                 *(answer -> text + i) = key_val;
17             }
18         }
19
20         if (hasCorrect == 0) {
21             stroke += 1;
22         }
23     }
24     wclear(IO_win);
25     box(IO_win, ACS_VLINE, ACS_HLINE);
26     wmove(IO_win, int(LINES / 2), 5);
27     wprintw(IO_win, answer -> text);
28     wmove(IO_win, int(LINES / 2) + 2, 5);
29     wprintw(IO_win, "Hint:");
30     wmove(IO_win, int(LINES / 2) + 3, 5);
31     wprintw(IO_win, hint -> text);
32     wrefresh(IO_win);
33     draw_man();
34
35     if (stroke >= 10) {
36         // dead
37         return -1;
38     } else if (strcmp(answer -> text, word -> text, word
39                 -> len) == 1) {
40         return 1;
41     }
42     return 0;
43 }

```

q_input.cpp

```

1  /**
2  * Function: q_input
3  * Description: input of question
4  * Parameter: key_val
5  * Return: 0 not decide or 1 decided
6  */
7
8  #include "hanged_man.h"
9

```

```

10 int hangedMan::q_input(int key_val) {
11     wclear(main_win);
12     box(main_win, ACS_VLINE, ACS_HLINE);
13     if (key_val == 9) {
14         q_input_stage += 1;
15         if (q_input_stage > 3) {
16             q_input_stage = 0;
17         }
18     } else {
19         switch (q_input_stage) {
20             case 0: // hardness
21                 switch (key_val) {
22                     case 49:
23                         hardness = 1;
24                         break;
25                     case 50:
26                         hardness = 2;
27                         break;
28                     case 51:
29                         hardness = 3;
30                         break;
31                 }
32                 break;
33
34             case 1: // word
35                 if (key_val >= 97 && key_val <= 122) {
36                     if (word -> len < WORDMAXLEN) {
37                         *(word -> text + word -> len) =
38                             key_val;
39                         word -> len += 1;
40                     }
41                 } else if (key_val == 127) { // delete
42                     if (word -> len > 0) {
43                         *(word -> text + word -> len - 1)
44                             = 0;
45                         word -> len -= 1;
46                     }
47                 }
48                 break;
49
50             case 2: // Hint
51                 if (key_val >= 97 && key_val <= 122) {
52                     if (hint -> len < HINTMAXLEN) {
53                         *(hint -> text + hint -> len) =
54                             key_val;
55                         hint -> len += 1;

```

```

53         }
54     } else if (key_val == 127) { // delete
55         if (hint -> len > 0) {
56             *(hint -> text + hint -> len - 1)
57                 = 0;
58             hint -> len -= 1;
59         }
60     }
61     break;
62
63     case 3:
64         if (key_val == 10) {
65             return 1;
66         }
67
68     default:
69         break;
70 }
71
72 // show
73 wmove(main_win, 1, 10);
74 wprintw(main_win, "Hardness");
75
76 wmove(main_win, 2, 10);
77 switch (hardness) {
78     case 1:
79         wprintw(main_win, "SIMPLE");
80         break;
81     case 2:
82         wprintw(main_win, "MIDDLE");
83         break;
84     case 3:
85         wprintw(main_win, "HARD");
86         break;
87     default:
88         break;
89 }
90
91 wmove(main_win, 3, 10);
92 wprintw(main_win, "Word");
93
94 wmove(main_win, 4, 10);
95 wprintw(main_win, word -> text);
96
97 wmove(main_win, 5, 10);

```

```

98     wprintw(main_win, "Hint");
99
100    wmove(main_win, 6, 10);
101    wprintw(main_win, hint -> text);
102
103    wmove(main_win, 7, 10);
104    wprintw(main_win, "Finish");
105
106    wmove(main_win, int(q_input_stage * 2 + 1), 1);
107    wprintw(main_win, "->");
108    return 0;
109 }

```

rst_input.cpp

```

1  /**
2  * Function: rst_input
3  * Description: get user's restart or quit input after
4  *               game end
5  * Parameter: key_val
6  * Return: 0: not decide, 1: decided
7  */
8  #include "hanged_man.h"
9
10 int hangedMan::rst_input(int key_val) {
11     int rt = 0;
12     switch (key_val) {
13         case 9:
14             restart += 1;
15             if (restart > 1) {
16                 restart = 0;
17             }
18             rt = 0;
19             break;
20         case 10:
21             rt = 1;
22             break;
23         default:
24             rt = 0;
25             break;
26     }
27
28     wclear(main_win);
29     wmove(main_win, int(LINES / 2) + 5, int(COLS / 2));
30     wprintw(main_win, "RESTART");

```

```

31     wmove(main_win, int(LINES / 2) + 6, int(COLS / 2));
32     wprintw(main_win, "QUIT");
33     wmove(main_win, int(LINES / 2) + 6 - restart, int(
34         COLS / 2) - 5);
35     wprintw(main_win, "-->");
36     wrefresh(main_win);
37     return rt;
38 }

```

draw_man.cpp

```

1  /**
2  * Function: draw_man
3  * Description: draw the man
4  * Parameter: None
5  * Return: None
6  */
7
8  #include "hanged_man.h"
9
10 void hangedMan::draw_man() {
11     wclear(man_win);
12     box(man_win, ACS_VLINE, ACS_HLINE);
13     // wprintw(man_win, "%d", stroke);
14     int counter = 0;
15     if (counter >= stroke) {
16         wrefresh(man_win);
17         return;
18     }
19
20     wmove(man_win, LINES - 5, 5);
21     whline(man_win, 42, 10);
22
23     counter += 1;
24     if (counter >= stroke) {
25         wrefresh(man_win);
26         return;
27     }
28
29     wmove(man_win, 5, 5);
30     wvline(man_win, 42, (LINES - 10));
31
32     counter += 1;
33     if (counter >= stroke) {
34         wrefresh(man_win);

```



```

35         return;
36     }
37
38     whline(man_win, 42, 8);
39
40     counter += 1;
41     if (counter >= stroke) {
42         wrefresh(man_win);
43         return;
44     }
45
46     wmove(man_win, 5, 13);
47     wvline(man_win, 42, 4);
48
49     counter += 1;
50     if (counter >= stroke) {
51         wrefresh(man_win);
52         return;
53     }
54
55     wmove(man_win, 7, 12);
56     whline(man_win, 42, 3);
57     wvline(man_win, 42, 3);
58     wmove(man_win, 9, 12);
59     whline(man_win, 42, 3);
60     wmove(man_win, 7, 14);
61     wvline(man_win, 42, 3);
62
63
64     counter += 1;
65     if (counter >= stroke) {
66         wrefresh(man_win);
67         return;
68     }
69
70     wmove(man_win, 8, 13);
71     wvline(man_win, 42, 4);
72
73
74     counter += 1;
75     if (counter >= stroke) {
76         wrefresh(man_win);
77         return;
78     }
79
80     wmove(man_win, 10, 10);

```

```

81     whline(man_win, 42, 3);
82
83     counter += 1;
84     if (counter >= stroke) {
85         wrefresh(man_win);
86         return;
87     }
88
89     wmove(man_win, 10, 13);
90     whline(man_win, 42, 4);
91
92     counter += 1;
93     if (counter >= stroke) {
94         wrefresh(man_win);
95         return;
96     }
97
98     wmove(man_win, 12, 12);
99     wvline(man_win, 42, 2);
100
101     counter += 1;
102     if (counter >= stroke) {
103         wrefresh(man_win);
104         return;
105     }
106
107     wmove(man_win, 12, 14);
108     wvline(man_win, 42, 2);
109     counter += 1;
110     if (counter >= stroke) {
111         wrefresh(man_win);
112         return;
113     }
114 }

```

win.cpp

```

1  /**
2  * Function: win
3  * Description: called after player2 win
4  * Parameter: None
5  * Return: None
6  */
7
8  #include "hanged_man.h"
9

```

```

10 void hangedMan::win() {
11     wclear(main_win);
12     wmove(main_win, int(LINES / 2), int(COLS / 2) - 5);
13     wprintw(main_win, "YOU WIN");
14     box(main_win, ACS_VLINE, ACS_HLINE);
15     wrefresh(main_win);
16 }

```

die.cpp

```

1  /**
2  * Function: die
3  * Description: what to do after die
4  * Parameter: None
5  * Return: None
6  */
7
8  #include "hanged_man.h"
9
10 void hangedMan::die() {
11     wclear(IO_win);
12     wmove(IO_win, int(LINES / 2), int(COLS / 4));
13     wprintw(IO_win, "YOU DIE!");
14
15     stroke = 5;
16     draw_man();
17
18     wmove(man_win, 13, 13);
19     wvline(man_win, 42, 4);
20
21
22     wmove(man_win, 15, 10);
23     whline(man_win, 42, 3);
24
25
26     wmove(man_win, 15, 13);
27     whline(man_win, 42, 4);
28
29
30     wmove(man_win, 17, 12);
31     wvline(man_win, 42, 2);
32
33     wmove(man_win, 17, 14);
34     wvline(man_win, 42, 2);
35
36     box(IO_win, ACS_VLINE, ACS_HLINE);

```

```

37     wrefresh(man_win);
38     wrefresh(IO_win);
39 }

```

del.cpp

```

1  /**
2  * Function: del
3  * Description: free the allocated memory
4  * Parameter: None
5  * Return: None
6  */
7
8  #include "hanged_man.h"
9
10 void hangedMan::del() {
11     endwin();
12     free_str(word);
13     free_str(hint);
14     free_str(answer);
15 }

```

init_free_str.cpp

```

1  #include "hanged_man.h"
2
3  /**
4  * Function: init_str
5  * Description: initizlize a string structure
6  * Parameter: string max length, initialize character
7  * Return: string* of the string.
8  */
9  string* hangedMan::init_str(int len, char chr) {
10     string* str;
11     str = (string*)malloc(sizeof(string));
12     str -> text = (char*)malloc(sizeof(char) * len);
13     for (int i = 0; i < len; i++) {
14         *(str -> text + i) = chr;
15     }
16     str -> len = 0;
17     return str;
18 }
19
20 /**
21 * Function: free_str
22 * Description: free the string structure
23 * Parameter: string* string

```

```

24     * Return: None
25 */
26 void hangedMan::free_str(string* str) {
27     free(str -> text);
28     free(str);
29 }

```

strcmp.cpp

```

1  /**
2  * Function: strcmp
3  * Description: use to compare two string which have same
   length
4  * Parameter: string1, string2, length
5  * Return: 0 not same or 1 same
6  */
7
8  #include "hanged_man.h"
9
10 int hangedMan::strcmp(char* s1, char* s2, int len) {
11     int same = 1;
12     for (int i = 0; i < len; i++) {
13         if (*(s1 + i) != *(s2 + i)) {
14             same = 0;
15         }
16     }
17     return same;
18 }

```

6 Testing of implementation

Compile environment

Mac OS X 10.5, g++ version 4.2.1, with curses lib installed

Makefile

```
1 main:
2 g++ -o hanged_man.out main.cpp init.cpp run.cpp draw_man.
  cpp a_input.cpp q_input.cpp strcomp.cpp die.cpp win.
  cpp del.cpp rst_input.cpp init_free_str.cpp -l curses
```

Running environment

Mac OS X 10.15, terminal window with size 80 cols and 24 rows

Testing results

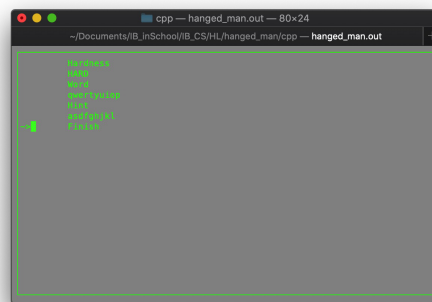


Figure 8: Player 1 input hardness, word, and hint



Figure 9: Player 2 input answer

