Tutorial 4: Jeopardy

Faculty of Engineering and Applied Science

SOFE 3950U : Operating Systems | CRN: 74171

Due: February 19th, 2024

Group 8

Daniel Amasowomwan [100787640]
daniel.amasowomwan@ontariotechu.net

Stanley Watemi [100648403]
stanley.watemi@ontariotechu.net

Fayomi Toyin [100765921]
oluwatoyin.fayomi@ontariotechu.net

Jeopardy Files

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>
#include "questions.h"
#include "players.h"
#include "jeopardy.h

#define BUFFER_LEN 256
#define NUM_PLAYERS 4




void tokenize(char *input, char **tokens);

int winner()

void show_results(player *players, int num_players);

int main(int argc, char *argv[])
    // Array of 4 players
    player players[NUM_PLAYERS+1];
    int test = 0;
    int rem_questions = NUM_OF_QUESTION-1;

    // Input buffer and and commands
    char buffer[BUFFER_LEN] = { 0 };
    char *tokens[BUFFER_LEN] = { 0 };

    // Display the game introduction and questions
    initialize_game();
    printf("Welcome to Jeopardy!\n");
    printf("Please enter the names of the four players\n");

    for(int i=1; i<=NUM_PLAYERS; i++)
```

```
36      for(int i=1; i<=NUM_PLAYERS; i++)
37      {
38
39          printf("Player %d?\t", i);
40          fgets(players[i].name, BUFFER_LEN, stdin);
41          if (players[i].name[strlen(players[i].name)-1] == '\n')
42          {
43              players[i].name[strlen(players[i].name)-1] = '\0';
44              }
45            players[i].score = 0; // initialise score to 0
46
47      }
48
49      system("clear"); //clear screen
50
51      char cat[BUFFER_LEN] = { 0 }; //question category
52      int value = 0;
53      char *token;
54      int player = 1;
55      int j = -1;
56
57      while (1) //actual game part
58      {
59          j = -1;
60          test = 0;
61          value = 0;
62          printf("%s? \n", players[player].name);
63          display_categories();
64          fgets(buffer, BUFFER_LEN, stdin);
65          buffer[strlen(buffer)-1] = '\0';
66
67      for (int i=0; i<strlen(buffer); i++)
68      {
69          if (buffer[i] == ' ')
70          {
```

```
70          {
71              test++;
72          }
73      }
74
75      if (test == 1)
76      {
77          token = strtok(buffer, " ");
78          strcpy(cat, token);
79          token = strtok(NULL, " ");
80          if (atoi(token))    value = atoi(token);
81
82      }
83          for(int jj = 0; jj < NUM_CATEGORIES; jj++)
84          {
85
86              if ( (strcmp(cat, categories[jj]) == 0) && (value == 100 ||
87               value == 200 || value == 300 || value == 400 ) ){
88                  if ( already_answered(&categories[jj], value) )
89                  { continue;}
90
91                  printf("I hope you have knowledge on %s. We've got %d on the line.\n",
92                  categories[jj], value);
93                  j = jj;
94
95              }
96
97          }
98          if (j == -1)
99          {
00              printf("Can you come again? I didn't get that.\n");
01              continue;
02          }
03
04
```

```
 98          if (j == -1)
 99          {
100              printf("Can you come again? I didn't get that.\n");
101              continue;
102          }
103
104          display_question(&categories[j], value);
105          fgets(buffer, BUFFER_LEN, stdin);
106          buffer[strlen(buffer)-1] = '\0';
107
108          if (valid_answer(&categories[j], value, buffer))
109          {
110                      if (value == 100)
111                          printf("Aamzing.\n");
112                      else if (value == 200)
113                              printf("Good Job.\n");
114                      else if (value == 300)
115                              printf("Exceptional.\n");
116                      else if (value == 400)
117                              printf("Good Job.\n");
118                      
119                      update_score(players, player, value);
120          }
121
122          else
123          {
124                      if (value == 100)
125                          printf("Incorrect?\n");
126                      else if (value == 200)
127                              printf("Wrong.\n");
128                      else if (value == 300)
129                              printf("Try again. Not.\n");
130                      else if (value == 400)
131                              printf("wrong again .\n");
132
133          }
```

```
133         }
134
135         if (rem_questions)
136         {
137                     rem_questions--;
138                     //printf("Questions left: %d\n", rem_questions);
139             if (player == NUM_PLAYERS)
140                     {
141                         printf("show them the leaderboard: \n");
142                         show_results(players);
143                         printf("New Round.");
144                         fgets(buffer, BUFFER_LEN, stdin); //wait for enter
145                        player = 1;
146                     }
147
148             else
149                     {
150                         printf("we have %s with %d points\n", players[player].name,
151                          players[player].score);
152                         printf(" next player");
153                        player += 1;
154                         fgets(buffer, BUFFER_LEN, stdin); //pause
155                     }
156
157             system("clear");//clear screen
158
159                     }
160
161         else //if we run out of questions
162                 {
163                 printf("Game over, the winner is %s with %d points. Congratulations.\n",
164                 players[winner(players)].name, players[winner(players)].score);
165                 return EXIT_SUCCESS;
166                 }
167
```

```
174 int winner(player1 *players)
175 {
176     int max = 0;
177         int best = 0;
178         int tie = 0;
179
180     for (int i = 1; i <=NUM_PLAYERS; i++)
181         {
182             if (players[i].score > max) //if player score is bigger than max...
183             {
184             max = players[i].score; //...set max to player score
185             best = i; //and set player number to best
186             }
187             if (players[i].score == max){
188                 tie = i;
189                 }
190
191         }
192
193         return best;
194 }
195
196 void show_results(player1 *players)
197 {
198
199     for (int i =1; i<=NUM_PLAYERS; i++)
200     {
201         printf("%s: %-5d", players[i].name, players[i].score);
202     }
203     printf("\n");
204
205 }
206
```

```c
void show_results(player1 *players)
{

    for (int i =1; i<=NUM_PLAYERS; i++)
    {
        printf("%s: %-5d", players[i].name, players[i].score);
    }
    printf("\n");

}

void tokenize(char *input, char **token)
{
        if (input[strlen(input)-1] == '\n') { input[strlen(input)-1] = '\0'; }
        char *p = strtok(input, token);
        while(p != NULL) {
            printf("%s\n", p);
            p = strtok(NULL, token);
        }
}
```

Question.c

```c
// Initializes the array of questions for the game
void initialize_game(void)
{
    // initialize the questions struct and assign it to the questions array
     printf("Welcome to jeopardy.\n"
         "Quick rundown of the rules: \n"
         "1. To select a quetion, type the category and the value .\n"
         "2. Answer in all caps, in 'WHO IS' or 'WHAT IS'. We really don't care which.\n"
         ;

}

// Displays each of the remaining questions and categories that have not been answered
void display_categories(void)
{
    // print categories and dollar values
    printf("We have\n");
    for (int i = 0; i < NUM_CATEGORIES; i++)
    {
            printf ("%-5s %-5s\t", categories[i], " ");

                for (int j = 0; j < 4; j++)
                {
                    if (quest[i*4 + j].answered == false)
                    {
                        printf("%-2d \t", quest[i*4 +j].value);
                    }
                            else printf("%-2s\t", " ");
                }

```

```
141 // Displays the question for the category and dollar value
142 void display_question(char *category, int value)
143 {
144    //printf("%s  %d\n", category, value);
145     for (int i=0; i<NUM_QUESTIONS; i++)
146     {
147         if ( ( strcmp(quest[i].category, category) == 0 ) && (quest[i].value == value) )
148         {
149         printf("\033[1;34m\t\t%s\033[0m\n", quest[i].question);
150         quest[i].answered = true;
151         break;
152         }
153     }
154 }
155
156
157 bool valid_answer(char *category, int value, char *answer)
158 {
159     int test = 0;
160     if (answer[strlen(answer)-1] == '\n') { answer[strlen(answer)-1] = '\0'; }
161     for (int i=0; i<strlen(answer); i++)
162     {
163         if (answer[i] == ' ')
164         {
165             test++;
166         }
167     }
168
169     if (! (test == 2) )
170     return false;
171         char *p = strtok(answer, " ");
172     if ( !(strcmp(&p, "WHO") || strcmp(&p, "WHAT") ) )
173     return false;
174     if (p != NULL)  p = strtok(NULL, " ");
175     if ( !strcmp(&p, "IS") ) return false;
176     if (p != NULL)  p = strt
```

```
180
181     for (int i=0; i<NUM_QUESTIONS; i++)
182     {
183         if ( ( strcmp(quest[i].category, category) == 0 ) && (quest[i].value == value) )
184         {
185             if ( strcmp(quest[i].answer, answer) == 0 )
186             {
187                 return true;
188             }
189             else { return false; }
190         }
191     }
192 }
193
194
195 bool already_answered(char *category, int value)
196 {
197     for (int i=0; i<NUM_QUESTIONS; i++)
198     {
199         if ( ( strcmp(quest[i].category, category) == 0 ) && (quest[i].value == value) )
200         {
201             if ( quest[i].answered == false )
202             {
203                 return false;
204             }
205             else { return true; }
206         }
207     }
208
```

Players

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>
#include "players.h"

// Returns true if the player name matches one of the existing players
bool player_exists(player *players, int num_players, char *name)
{
    for (int i = 0; i<num_players;i++)
    {
        if(strcmp(players[i].name,name)==0)
        {
            return true;
        }
    }
    return false;
}

// Go through the list of players and update the score for the player with the corresponding name
void update_score(player *players, int num_players, char *name, int score)
{
    for (int i = 0; i<num_players;i++)
    {
        if(strcmp(players[i].name,name)==0)
        {
            players[i].score = players[i].score + score;
        }
    }
}
```

[SOFE3950U-OPSYS-Tutorials/tut4/questions.h at main · Daniel-Amas/SOFE3950U-OPSYS-Tutorials (github.com)]