Faculty of Engineering and Applied Science

SOFE 3950U / CSCI 3020U: Operating Systems

Tutorial 4

Group Member 1

Name: Alexander Campbell

Student ID: 100703650

Group Member 2

Name: Atharshan Kennedy

Student ID: 100590243

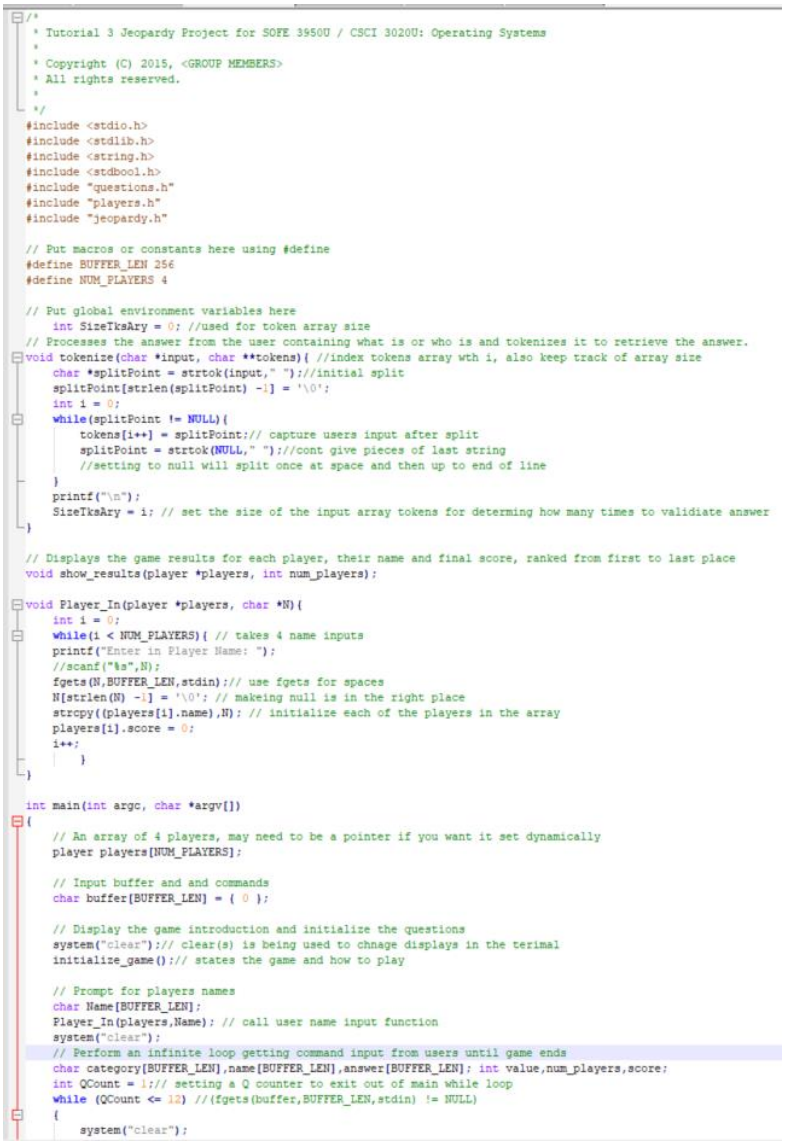
Group Member 3

Name: Joey Villafuerte

Student ID: 100759003

Date: February 20, 2021

Code Explanation



```
/*
 * Tutorial 3 Jeopardy Project for SOFE 3950U / CSCI 3020U: Operating Systems
 *
 * Copyright (C) 2015, <GROUP MEMBERS>
 * All rights reserved.
 *
 */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>
#include "questions.h"
#include "players.h"
#include "jeopardy.h"

// Put macros or constants here using #define
#define BUFFER_LEN 256
#define NUM_PLAYERS 4

// Put global environment variables here
    int SizeTksAry = 0; //used for token array size
// Processes the answer from the user containing what is or who is and tokenizes it to retrieve the answer.
void tokenize(char *input, char **tokens){ //index tokens array wth i, also keep track of array size
    char *splitPoint = strtok(input," ");//initial split
    splitPoint[strlen(splitPoint) -1] = '\0';
    int i = 0;
    while(splitPoint != NULL){
        tokens[i++] = splitPoint;// capture users input after split
        splitPoint = strtok(NULL," ");//cont give pieces of last string
        //setting to null will split once at space and then up to end of line
    }
    printf("\n");
    SizeTksAry = i; // set the size of the input array tokens for determing how many times to validiate answer
}

// Displays the game results for each player, their name and final score, ranked from first to last place
void show_results(player *players, int num_players);

void Player_In(player *players, char *N){
    int i = 0;
    while(i < NUM_PLAYERS){ // takes 4 name inputs
    printf("Enter in Player Name: ");
    //scanf("%s",N);
    fgets(N,BUFFER_LEN,stdin);// use fgets for spaces
    N[strlen(N) -1] = '\0'; // makeing null is in the right place
    strcpy((players[i].name),N); // initialize each of the players in the array
    players[i].score = 0;
    i++;
        }
}

int main(int argc, char *argv[])
{
    // An array of 4 players, may need to be a pointer if you want it set dynamically
    player players[NUM_PLAYERS];

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

    // Display the game introduction and initialize the questions
    system("clear");// clear(s) is being used to chnage displays in the terimal
    initialize_game();// states the game and how to play

    // Prompt for players names
    char Name[BUFFER_LEN];
    Player_In(players,Name); // call user name input function
    system("clear");
    // Perform an infinite loop getting command input from users until game ends
    char category[BUFFER_LEN],name[BUFFER_LEN],answer[BUFFER_LEN]; int value,num_players,score;
    int QCount = 1;// setting a Q counter to exit out of main while loop
    while (QCount <= 12) //(fgets(buffer,BUFFER_LEN,stdin) != NULL)
    {
        system("clear");
```

In Jeopardy.c file the player is promoted to enter in name. This is done through Player_In() function where names are entered in and score per person is set 0. This done four times

In the while loop the QCount is used to make sure that only 12 questions are answered. Then through the function display_categories() the questions categories and questions answered values are displayed to the players. Then players name and player number as well a score are placed underneath. And for when players are typing in their names they selected to play at random before hand.

When ever input is made the buffer must be replaced fully with null to avoid garbage being left behind from before. Also, when an input is made a null must be added at the correct place.

Player's name input is checked to see if it is valid while also seeing if it is the user that was selected. If passed the user playing is asked to select the category and value question. Checking is done to make user's input is made correctly.

Once the selection is done it is run through another check, already_answered() function to determine if the question selected is not answered. If answered then return and ask for a different selection. If not allow user to answer question. User input is then sent tockenize() function to create the input array for checking each of users inputs against the answer. The input answers are sent to the valid_answer() function. Here input per input is sent and once a true is sent back Y is set as true and exits loop. If Y is true, then correct answer is displayed. If not, then incorrect answer is displayed. If correct, then update score of the player through update_score() function. Then random select the next player and exit out of inner while loop. If users name was not correct initially then inner while loop will loop until name is correct.

```
{
    system("clear");
    display_categories();//call and create the dsiplay of question values
    for (int i = 0; i<NUM_PLAYERS ;i++){ printf("[Player %d: ",i+1); printf("%s $%d]",players[i].name,players[i].score);  }
    printf("\n"); // the for loop is used to print out Players name and scores at the bottom of the question values
    int num,lower = 1,upper = 5; // used for rand sel a player to type in their name
    num = rand()%(upper-lower)+1;
    char bufReplace[BUFFER_LEN];// used for clearing out garbage
    memset(bufReplace,'\0',sizeof(char)*BUFFER_LEN); // set the above with all nulls
    printf("Type in your name Player %d: ",num);// promt user to type in their name
    for (;;){
        strcpy(buffer,bufReplace);// replce the buffer array on line 58  to get rid of garbage
        fgets(buffer,BUFFER_LEN,stdin);// get teh user input
        buffer[strlen(buffer) -1] = '\0'; // make sure that null is in the right place
        if (player_exists(players, NUM_PLAYERS, buffer) == 1 && strcmp(players[num-1].name,buffer) == 0){
            system("clear"); // make sure the the player inputed is the player playing
            display_categories(); // redisplay with user playing
            printf("\nPlayer %d please type a Category <CATEGORY OPTIONS:  P,A,D > " ,num);
            while(1){ // ask for what category that will want to choose then he value from that category
                for(;;){ // infinte for loops used for makign sure the sure sel correctly
                    fgets(category,BUFFER_LEN,stdin); // get user input and make sure it is correct
                    category[strlen(category) - 1] = '\0';
                    category[0] = toupper(category[0]);
                    if(strcmp(category,"P")==0 || strcmp(category,"A")==0 || strcmp(category,"D")==0 ){
                        break;
                    }else{printf("\nPlease type a category P,A,D: ");}
                    strcpy(category,bufReplace);
                }
                printf("\nPlayer %d please type a value from category %s > " ,num,category);
                for(;;){
                    char strInt[BUFFER_LEN]; // gettign user value as a string teh using atoi to make it int
                    //scanf("%d",&value);    // from tthere the int is used to check for right val sel
                    strcpy(buffer,bufReplace);
                    fgets(strInt,BUFFER_LEN,stdin);
                    strInt[strlen(strInt) -1] = '\0';
                    value = atoi(strInt);
                    if (value == 400 || value == 800 || value == 1200 || value == 1600){
                        break;
                    }else{printf("\nPlease type in a number that listed above for category %s: ", category);}
                }
                if(already_answered(category, value) == false){ break; // make sure that user sel a not answered question
                } else {
                printf("\nSry that has already been taken, Player %d please type a Category <CATEGORY OPTIONS:  P,A,D > " ,num);
                }
            }
            fflush(stdin); // flush out left over input
            display_question(category, value); // display teh question teh user selected
            printf("\nWhat is your answer Player %d :", num); // Ask for answerfrom user
            strcpy(buffer,bufReplace);
            fgets(buffer,BUFFER_LEN,stdin); // get user input and make sure that the null is in the right place
            buffer[strlen(buffer) -1] = '\0';
            char *array[BUFFER_LEN + 1]; // allocated memeory for pointer to point
            char **Tks = array;// holding pieces of text for input with space
            tokenize(buffer,Tks);// call tokenizen for distroying user input and creating a array of inputs
            int t = 0; bool Y = false; // use a bool to determine when a answer is correct and conitne otherwise
                                       // with checikng until the end of the array of inputs
            for(t = 0; t < SizeTksAry ;t++){
                if (valid_answer(category, value, Tks[t]) == false){continue;}
                else{Y = true; break;}
            }
            if (Y == false){system("clear");printf("\n---SORRY THAT'S INCORRECT---");}
            else {  // when Y is false the user did not get teh correct answer, when true
                system("clear"); // proceed to updating player socre
                printf("\n!!!CORRECT!!!");
                update_score(players,NUM_PLAYERS, players[num-1].name,value);
            }
            printf("\n");
            sleep(1);
            num = rand()%(upper-lower)+1;// used for getting teh player number
            break;// break to allow for the next player to play
        // Call functions from the questions and players source files
```

Once the outer while loop has finished the results are printed out. (Not working but here's the idea for sorting). Find max and record it with the max index.

The max index is placed in indexs array.

```
            break;// break to allow for the next player to play
        // Call functions from the questions and players source files

        // Execute the game until all questions are answered

        // Display the final results and exit
        } else { printf("Name Inncorrect please try again Player %d: ",num);}// this outputs when teh user does not input a name corretcly
    }
    QCount++;// counter teh Q count for exiting loop when 12 questions have reached
}
system("clear");// clear for the end results
printf("\n");
printf("\nEND GAME RESULTS:\n"); // end ruelts do bot work
int indexs[4];
int k  = 0 ;
while(k < 4){ // used for finding the max and printing it out whil also remeber not to check that number
    int max = 0,ind = 0;
    bool x = false;
    for(int i = 0;i<4;i++){ // loop each player
            for (int j;j<4;j++){ // if players score index is in indexs then skip doign any checks with this index
                if (i == indexs[j]){x = true; break;}
            }
            if ( x == false && max < players[i].score){max = players[i].score; ind = i;} //  used for replacing the
                                                    // current max and max index
    }
    indexs[k] = ind;// addign max index to indexs array for ignoring in other checks
    k++;
}
for (int i = 0;i<4;i++){
    printf("\n%s %d",players[indexs[i]].name,players[indexs[i]].score);// printing out the player and user score
}
printf("\n");
return EXIT_SUCCESS;
}
```

Then loop again this time checking if the numbers index has already been checked for max, which then will result in no checking when a numbers index that was previously checked for max was found in the indexs array. This will go from all four numbers checking in the for loop to just one number left, who's indx was not recorded in the indexs array for acceding score values.

```
/*
 * Tutorial 3 Jeopardy Project for SOFE 3950U / CSCI 3020U: Operating Systems
 *
 * Copyright (C) 2015, <GROUP MEMBERS>
 * All rights reserved.
 *
 */
#ifndef JEOPARDY_H_
#define JEOPARDY_H_

#define MAX_LEN 256

// Processes the answer from the user containing what is or who is and tokenizes it to retrieve the answer.
extern void tokenize(char *input, char **tokens);

// Displays the game results for each player, their name and final score, ranked from first to last place
extern void show_results(player *players, int num_players);

extern void Plare_In(player *players, char N);// Added in for player name input

#endif /* JEOPARDY_H_ */
```

The header for the Jeopardy.c file is used for allowing void functions from the source file. This will allow for externally access from other files. Also, the Player_In is added as well.

```c
/*
 * Tutorial 3 Jeopardy Project for SOFE 3950U / CSCI 3020U: Operating Systems
 *
 * Copyright (C) 2015, <GROUP MEMBERS>
 * All rights reserved.
 *
 */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "questions.h"
#define MAX_LEN 256
#define NUM_QUESTIONS 12

// Initializes the array of questions for the game
void initialize_game(void){
    printf("Welcome To Jeopardy\n"); // Beginning statments // VV Questions,Answers,Value bank
    printf("Please type in all four players names to proceed: \n");
    char Q[NUM_QUESTIONS][MAX_LEN] = {"What is Programming?","What is an int?","What is system.out.println?","What is .exe",
                                      "Who is Alan Turing?","What is O<n^2>?","What is merger sort?","What is upperbound?",
                                      "Who is created db?","What is a db?","What is a db sofware?","What is a server?"};
    char A[NUM_QUESTIONS][MAX_LEN] = {"coding","int","printing","excutable",
                                      "mathematician","quadractic","divide","BigTheta",
                                      "Edgar","collection","mysql","Network"};
    int s[] = {400,800,1200,1600,400,800,1200,1600,400,800,1200,1600,400,800,1200,1600};
    int j = 0 , i = 0; // i for categoriesindexing and j for question indexing
    while( j < 12 ){ // switch i when j reaches a max of four quetsions and exit while when j = 12
        if (j%4 == 0 && j != 0){ // increamnet i when j is a mul of 4 and not 0 to avoid errors
        i++;
        }
        strcpy(questions[j].category,categories[i]);
        strcpy(questions[j].question,Q[j]);
        strcpy(questions[j].answer,A[j]);
        questions[j].value = s[j];
        questions[j].answered = false;
        j++;
    }
    // initialize each question struct and assign it to the questions array
}

// Displays each of the remaining categories and question dollar values that have not been answered
void display_categories(void)
{
    printf("                                    JEOPARDY\n"); // Game layout
    printf("Radomly selected player types name in  then chooses a question value to answer: \n");
    printf("Question Board: \n");
    int mul = 0;   // uses this for the inner loop when changing categories to display q values for each categories
    for(int i = 0;i<3;i++){
        printf("%s >",categories[i]);
        for(int j = 0 + mul; j<4+mul ;j++){
            if (questions[j].answered == false){
                printf("[ $%d ]",questions[j].value);
            }
        }
        mul = mul + 4;
        printf("\n");
    }
    printf("\n");
    // print categories and dollar values for each unanswered question in questions array
}

// Displays the question for the category and dollar value
void display_question(char *category, int value){
    int offset = 0; // use switch statment  to determing where to offset i for displaying the sel question
    switch(category[0]){ case 'A': offset = 4; break; case 'D': offset = 8; break; default: break;}
    for( int i = offset; i < (offset + 4);i++){
        if (questions[i].value == value){  //out put category,question,value to user
            printf("\n%s: %s--|value<%d>|: ",questions[i].category,questions[i].question,questions[i].value);
            break;
        }
    }
}
```

Next is the questions.c source file. Initializing_game void function is used to create the bank of questions and displaying how to play. Displaying_categories() is used to display the categories to select from and the question values that have not been answered.

Display_question() function is used to display a question, that was selected, that has not been answered to the player to be answered. For knowing if the questions are answered or not that is done with the last the void function of this file.

```
// Returns true if the answer is correct for the question for that category and dollar value
bool valid_answer(char *category, int value, char *answer){
    int offset = 0; // switch used for offset i when checking for valid answer form a category
    switch(category[0]){ case 'A': offset = 4; break; case 'D': offset = 8; break; default: break;}
    for( int i = offset; i < (offset + 4);i++){
        if (questions[i].value == value){ // Once question value and user value match for a category
            if (strcmp(questions[i].answer,answer) == 0){ // check if answers match
                questions[i].answered = true;  return true; // if so return true and set the question as true
            } else {questions[i].answered = true;} // if not still set question as true
        }       // when set true it's value will not be displayed next time when display_question() is called
    }
    // Look into string comparison functions
    return false;
}

// Returns true if the question has already been answered
bool already_answered(char *category, int value){
    int offset = 0;// use switch statment to offset i for seeing from a category of the question the user selected
                   // was already answered. If return false  and if not return true
                   // for allowing user to not proceed and choose another question or to proceed with the question
    switch(category[0]){ case 'A': offset = 4; break; case 'D': offset = 8; break; default: break;}
    for( int i = offset; i < (offset + 4);i++){
        if (questions[i].value == value){
            if (questions[i].answered == true){ return true;}
        }
    }
    // lookup the question and see if it's already been marked as answered
    return false;
}
```

The next void function is the valid_answer function where the answer is check and see if one of the user inputs is the answer. In both right and wrong the function will set true to the question being answered to not being asked again.

Finally, the last function is what allows the user selected question to be displayed or not. If the functions, see that true is placed for the question then it is not allowed to be displayed and therefore will result in a return false back to Jeopardy.c.

Next file is the question.h file. This where the questions array framework is made, categories is set, and the void function can be allowed for external use from other files. Struct is used for creating the array of questions, in struct the elements are laid out to what is needed to place into each index of questions array.

```
/*
 * Tutorial 3 Jeopardy Project for SOFE 3950U / CSCI 3020U: Operating Systems
 *
 * Copyright (C) 2015, <GROUP MEMBERS>
 * All rights reserved.
 *
 */
#ifndef QUESTIONS_H_
#define QUESTIONS_H_

#include <stdbool.h>

#define MAX_LEN 256
#define NUM_CATEGORIES 3
// The number of questions, you can use this in your functions in
// questions.c, this can be accessed in questions.c
#define NUM_QUESTIONS 12

// List of 3 categories as array of strings
static char categories[NUM_CATEGORIES][MAX_LEN] = {
    "programming",
    "algorithms",
    "databases"
};

// Questions struct for each question
typedef struct {
    char category[MAX_LEN];
    char question[MAX_LEN];
    char answer[MAX_LEN];
    int value;
    bool answered;
} question;

// An array of 12 questions (4 for each category), initialized in initialize_game
// this may need to be a pointer if you want it set dynamically
question questions[NUM_QUESTIONS];

// Initializes the array of questions for the game
extern void initialize_game(void);

// Displays each of the remaining categories and question dollar values that have not been answered
extern void display_categories(void);

// Displays the question for the category and dollar value
extern void display_question(char *category, int value);

// Returns true if the answer is correct for the question for that category and dollar value
extern bool valid_answer(char *category, int value, char *answer);

// Returns true if the question has already been answered
extern bool already_answered(char *category, int value);

#endif /* QUESTIONS_H_ */
```

```c
/*
 * Tutorial 3 Jeopardy Project for SOFE 3950U / CSCI 3020U: Operating Systems
 *
 * Copyright (C) 2015, <GROUP MEMBERS>
 * All rights reserved.
 *
 */
#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){
    int found = 0;    // using 0 or 1 for determing if a player was found or not
    for (int i = 0; i < num_players ;i++){ // loop and compare user enterned name to all names
        if (strcmp(players[i].name,name) == 0){ found = 1; break; } // once found break and found is 1;
    }
    return found; // return 0 for no and 1 for yes
    //if (found == 0){return 0;} else {return 1;}
}

// Go through the list of players and update the score for the
// player given their name
void update_score(player *players, int num_players, char *name, int score){
        for (int i = 0; i < num_players ;i++){ // for teh player that answered correctlyy
            if(players[i].name == name){// loop until the person is found and update their score
                players[i].score = players[i].score + score;
            }
        }
}
```

The next two files are the players.c and players.h.

Players.c has two functions, player_exists() and update_score().

Player_exists() checks user name input that came form when the player enters their name to make a selection. Here the name is check against all names to see if the name exists to allow for player to proceed with selection.

The last function will take players name, value that selected during selection procedure and as well the list of all players' name. The list will make sure to allow for the program to find the user and add the score to their current score.

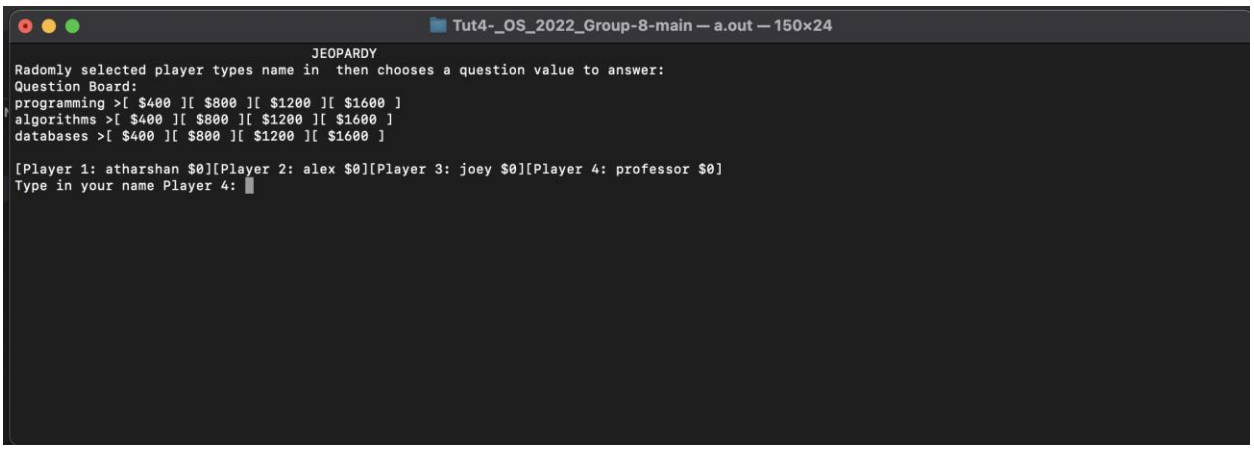
The last program file for the game is the player.h. Here , like other header files, the void functions from players.c are allowed to be used from other files. The struct for player information array framework is placed here as well. The struct helps store players identification and score that will be used later when the questions are being answered.

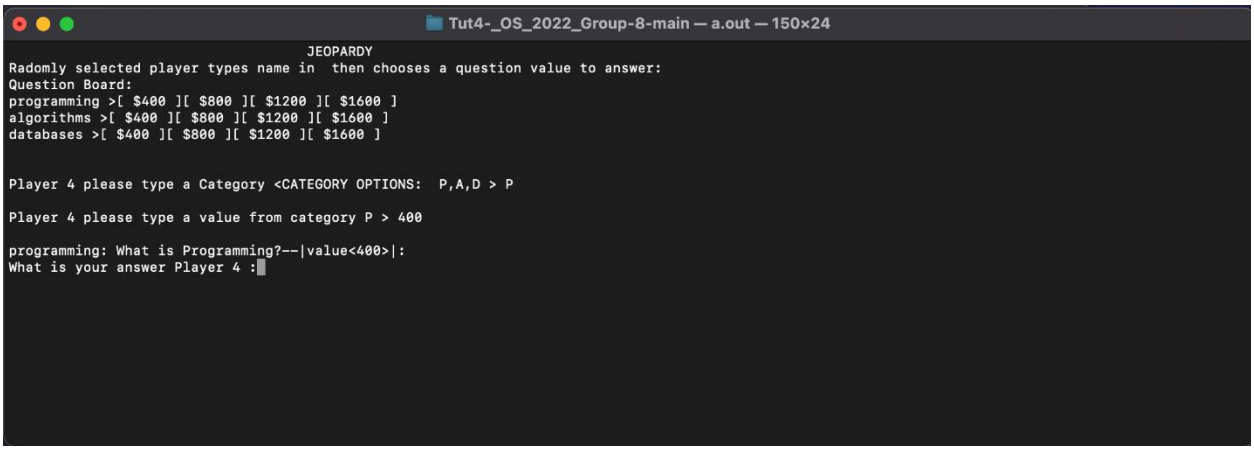
```c
/*
 * Tutorial 3 Jeopardy Project for SOFE 3950U / CSCI 3020U: Operating Systems
 *
 * Copyright (C) 2015, <GROUP MEMBERS>
 * All rights reserved.
 *
 */
#ifndef PLAYERS_H_
#define PLAYERS_H_

#include <stdbool.h>

#define MAX_LEN 256

// Player struct for each player
typedef struct {
    char name[MAX_LEN];
    int score;
} player;

// Returns true if the player name matches one of the existing players
extern bool player_exists(player *players, int num_players, char *name);

// Go through the list of players and update the score for the
// player given their name
extern void update_score(player *players, int num_players, char *name, int score);

#endif /* PLAYERS_H_ */
```

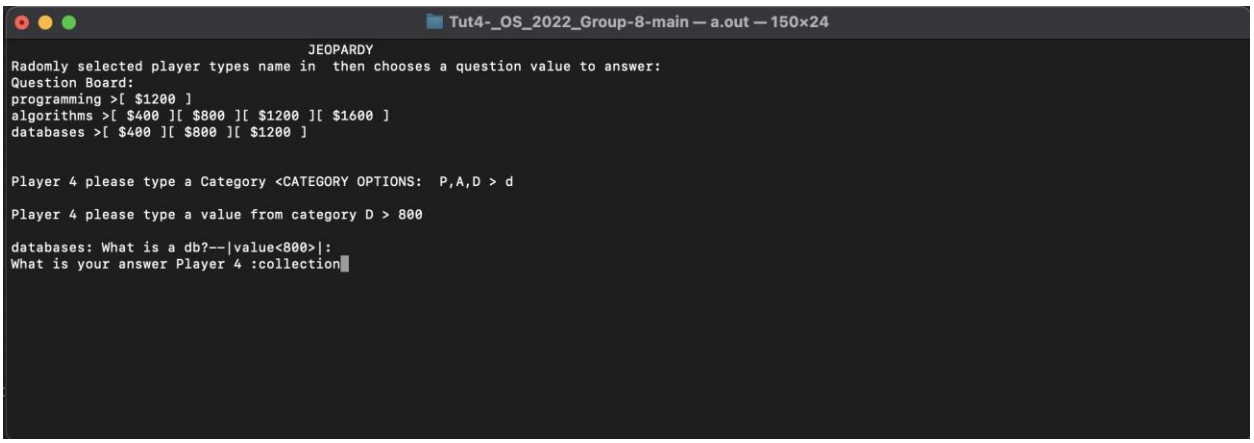
```
CC = gcc
CFLAGS = -Wall -Wextra -std=c99
LFLAGS =
LIBS =
SOURCES = jeopardy.c questions.c players.c
OBJECTS = $(subst .c,.o,$(SOURCES))
EXE = jeopardy.exe
.PHONY: clean help

jeopardy.exe : jeopardy.o questions.o players.o
	$(CC) $(CFLAGS) $^ $(LIBS) -o $@

%.o : %.c
	$(CC) $(CFLAGS) -c $<

all : $(EXE)

clean:
	rm -f $(OBJECTS) $(EXE) *~

cleanup:
	rm -f $(OBJECTS) *~

help:
	@echo "Valid targets:"
	@echo "  all:    generates all binary files"
	@echo "  clean:  removes .o and .exe files"
```

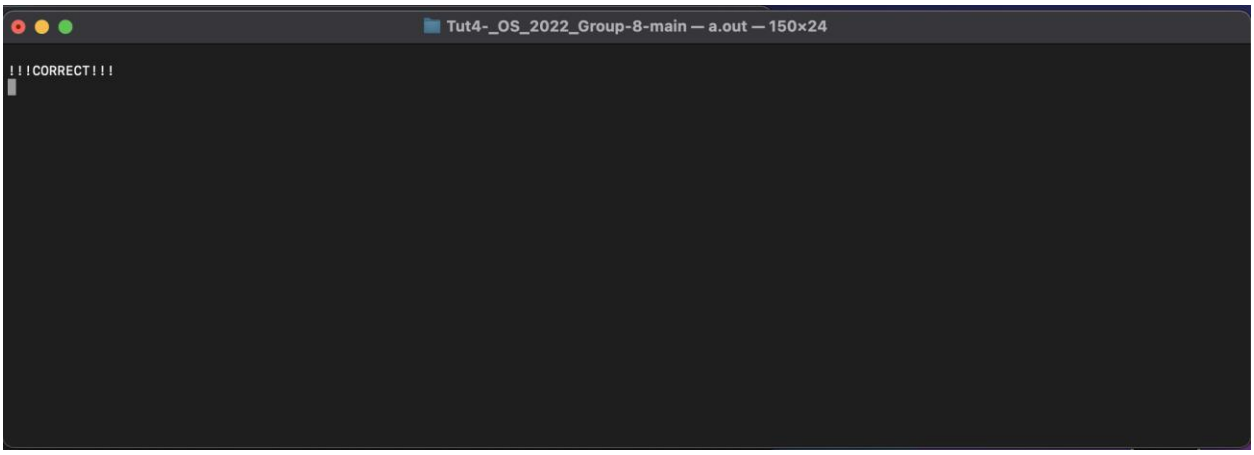
The last file is the Makefile where the Jeopardy.exe can be made. Nothing had to be altered here.

Sample Runs


Tut4-_OS_2022_Group-8-main — a.out — 150×24
JEOPARDY
Radomly selected player types name in then chooses a question value to answer:
Question Board:
programming >[$400][$800][$1200][$1600]
algorithms >[$400][$800][$1200][$1600]
databases >[$400][$800][$1200][$1600]
[Player 1: atharshan $0][Player 2: alex $0][Player 3: joey $0][Player 4: professor $0]
Type in your name Player 4:


Tut4-_OS_2022_Group-8-main — a.out — 150×24
JEOPARDY
Radomly selected player types name in then chooses a question value to answer:
Question Board:
programming >[$400][$800][$1200][$1600]
algorithms >[$400][$800][$1200][$1600]
databases >[$400][$800][$1200][$1600]
Player 4 please type a Category <CATEGORY OPTIONS: P,A,D > P
Player 4 please type a value from category P > 400
programming: What is Programming?--|value<400>|:
What is your answer Player 4 :


Tut4-_OS_2022_Group-8-main — a.out — 150×24
JEOPARDY
Radomly selected player types name in then chooses a question value to answer:
Question Board:
programming >[$1200]
algorithms >[$400][$800][$1200][$1600]
databases >[$400][$800][$1200]
Player 4 please type a Category <CATEGORY OPTIONS: P,A,D > d
Player 4 please type a value from category D > 800
databases: What is a db?--|value<800>|:
What is your answer Player 4 :collection

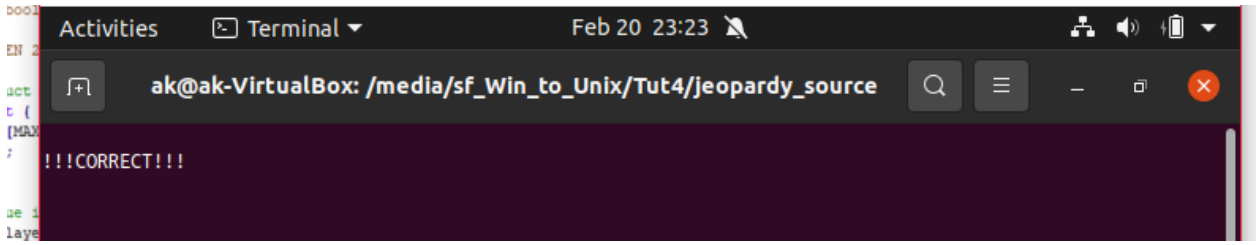



```
Welcome To Jeopardy
Please type in all four players names to proceed:
Enter in Player Name: A
Enter in Player Name: G
Enter in Player Name: H
Enter in Player Name: T
```

```
                                JEOPARDY
Radomly selected player types name in  then chooses a question value to answer:
Question Board:
programming >[ $400 ][ $800 ][ $1200 ][ $1600 ]
algorithms >[ $400 ][ $800 ][ $1200 ][ $1600 ]
databases >[ $400 ][ $800 ][ $1200 ][ $1600 ]

[Player 1: A $0][Player 2: G $0][Player 3: H $0][Player 4: T $0]
Type in your name Player 4:
```

```
                                JEOPARDY
Radomly selected player types name in  then chooses a question value to answer:
Question Board:
programming >[ $400 ][ $800 ][ $1200 ][ $1600 ]
algorithms >[ $400 ][ $800 ][ $1200 ][ $1600 ]
databases >[ $400 ][ $800 ][ $1200 ][ $1600 ]


Player 4 please type a Category <CATEGORY OPTIONS:  P,A,D > P

Player 4 please type a value from category P > 400

programming: What is Programming?--|value<400>|:
What is your answer Player 4 :
```

```
                                JEOPARDY
Radomly selected player types name in  then chooses a question value to answer:
Question Board:
programming >[ $400 ][ $800 ][ $1200 ][ $1600 ]
algorithms >[ $400 ][ $800 ][ $1200 ][ $1600 ]
databases >[ $400 ][ $800 ][ $1200 ][ $1600 ]


Player 4 please type a Category <CATEGORY OPTIONS:  P,A,D > p

Player 4 please type a value from category P > 400

programming: What is Programming?--|value<400>|:
What is your answer Player 4 :h coding
```




```
                                JEOPARDY
Radomly selected player types name in  then chooses a question value to answer:
Question Board:
programming >[ $800 ][ $1200 ][ $1600 ]
algorithms >[ $400 ][ $800 ][ $1200 ][ $1600 ]
databases >[ $400 ][ $800 ][ $1200 ][ $1600 ]

[Player 1: A $0][Player 2: G $0][Player 3: H $0][Player 4: T $400]
Type in your name Player 2:
```