```
//

/*

Group Number: E

▪ Group member name: Krish Ravi

▪ Email: vignesh.ravi@okstate.edu

▪ Date: 03/04/2022

Responsibilities: Exclusively Game Logic


Completed: Entirety of actual game logic alongside Adam Loeckle, including pass
and reset implementation

           Entire game was possible to play purely locally with all intended
features, this was done as Krish finished his share before Server-client
architecture was in place,

                to allow for easier later integration.

           Collaboration with final server and client architectures, ensuring
game control flow is maintained


References

-> https://stackoverflow.com/questions/63132911/check-if-a-string-is-included-in-
an-array-and-append-if-not-c

-> https://www.efaculty.in/c-programs/check-whether-a-given-word-exists-in-a-
file-or-not-program-in-c/

-> https://stackoverflow.com/questions/19429138/append-to-the-end-of-a-file-in-c

*/


#include "Main.h"



int computerTurn()

{

    //Computer will check input file line by line for usable words (if they're
wordbuilder words that haven't been used yet, it plays them)

    int prevlen = 0;
```

```
for (int i = 0; prev[i]!='\0'; i++)
{
    prevlen++;
}
FILE* filePointer;
int wordExist=0;
int bufferLength = 99;
char line[bufferLength];
int linelen = 0;
int disallowed = 0;
int run = 1;
int skip = 0;
strcpy(newf,""); //"new\n"
strcpy(newadd,"\n");    // "\nnew\n"
filePointer = fopen(fname, "r");
while(fgets(line, bufferLength, filePointer) && run!=0)
{
    if (skip!=3)
    {
        skip++;
        continue;
    }
    disallowed = 0;
    strcpy(new,"");
    int c = getc(filePointer);
    if (c == EOF)
    {
        printf("\nComputer could not find appropriate word");
        return 0;
    }
```

```
else
{
    ungetc(c,filePointer);
}
linelen=0;
for (int i = 0; line[i]!='\0'; i++)
{
    linelen++;
}
for (int i = 0; i < linelen-1; i++)
{
    new[i] = line[i];
}
new[linelen-1] = '\0';
strcpy(newf,"");
strcpy(newadd,"\n");
strcat(newf, new);
//strcat(newf,"\n");
strcat(newadd, new);
strcat(newadd,"\n");
size_t n = sizeof(prev)/sizeof(char);
nnew = sizeof(new)/sizeof(char);
size_t nnewf = sizeof(newf)/sizeof(char);
for (int i=0; i<n;i++)
{
    for (int x = 0; x < nnew && disallowed==0 && new[x]!='\0'; x++)
    {
        //make sure no disallowed characters are in it
        for (int y = 0; y < 6 && new[x]!='\0'; y++)
        {
```

```
                    //printf("\n Iteration %d y iteration %d we're looking at %c
in new and %c in letters\n", x, y, new[x], letters[y]);
                    if (new[x]!=letters[y])
                    {
                        if (letters[y+1]=='\0')
                            disallowed=1;
                        else
                            continue;
                    }
                    else
                        break;
                }
            }
            if (disallowed==0)
            {
                if (new[0]==prev[i])
                {
                    //printf("\nUsed correct characters!");
                    int j = i;
                    int k = 0;
                    while ((j<n) && (new[k]==prev[j]) && !((new[k]=='\0') &&
(prev[j]=='\0')))
                    {
                        //printf("\n Iteration %d we're looking at %c in new and
%c in prev\n", i, prev[j], new[k]);
                        j++;
                        k++;
                        //printf("The value of j is %d k is %d n is %d", j,k,n);
                    }
                    if ((j==n) || (prev[j]=='\0') || ((new[k]=='\0') &&
(prev[j]=='\0')))
                    {
                        //printf("\nComputer's Word is valid!");
                        //check if word has already been used reference
                        int dup = 0;
                        for (int j = 0; j < 100; j++)
                        {
                            if(strcmp(new, usedWords[j]) == 0)
                            {
                                printf("\nWord %10s has been found in %d of
usedWords as %10s",new,j,usedWords[j]);
                                dup = 1;   // got a duplicate
                            }
                        }
                        if (dup == 0) {    // not a duplicate: add it to
usedWords
                            strcpy(usedWords[noUsedWords+1], new);
                            noUsedWords += 1;
                        }
                        if(dup)
                        {
                            printf("\nWORD HAS ALREADY BEEN USED THIS GAME.");
                            //penalise
                            for (int i = 0; i<=noUsedWords;i++)
                            {
                                printf("\nUsed word %d of %d is
%s",i,noUsedWords,usedWords[i]);
                            }
                            break;
                        }
                        else
                        {
                            //printf("\nWord has NOT been used this game. Added
to used words.");
```

```
                            printf("\nComputer played the word: %s",new);
                            for (int i = 0; i<=noUsedWords;i++)
                            {
                                printf("\nUsed word %d of %d is
%s",i,noUsedWords,usedWords[i]);
                            }
                            strcpy(prev,new);
                            strcpy(new,"");
                            run = 0;
                            return 1;

                        }
                    }
                    else
                    {
                        if (j<n)
                            continue;
                        printf("\n Invalid but part of it was at some point");
                        //in theory we should never be here?
                        //penalise
                    }
                }
                if (i==(n-1))
                {
                    //printf("\nComputer's Word is not valid.");
                    //penalise
                }
            }
            else
            {
                printf("\nWord contains disallowed characters.");
                //penalise
            }
        }
    }
    fclose(filePointer);
}

void dictionaryCheck(mqd_t dictionary, size_t nnewf, char *lowernew, int
newSocket)
{
    for(int w = 0; w<nnewf; w++)
    {
        lowernew[w] = tolower(newf[w]);
    }
    FILE* filePointerd;
    int wordExistd=0;
    int bufferLengthd = 255;
    char lined[bufferLengthd];
    int linedlen = 0;
    int lowernewlen = 0;
    printf("\nChecking if %s is a valid dictionary word\n", lowernew);
    filePointerd = fopen("dictionary.txt", "r");
    for (int i = 0; lowernew[i]!='\0'; i++)
    {
        lowernewlen++;
    }
    while(fgets(lined, bufferLengthd, filePointerd))
    {
        linedlen=0;
        for (int i = 0; lined[i]!='\0'; i++)
        {
            linedlen++;
        }
        char *ptrd = strstr(lined, lowernew);
        if (ptrd != NULL && (linedlen==lowernewlen))
        {
            //printf("\nline is %d characters long and newf is %d
long",linedlen,lowernewlen);
            wordExistd=1;
            break;
        }
    }
    bzero(lowernew,sizeof(lowernew));
    fclose(filePointerd);
    if (wordExistd==1)
    {
        sendDictionaryMsg(dictionary, "CORRECT", 7);
    }
    else
    {
        sendDictionaryMsg(dictionary, "INCORRECT", 9);
    }
}

int inputCheck()
{
    //check if word is already in the input file
    FILE* filePointer;
    int wordExist=0;
    int bufferLength = 255;
    char line[bufferLength];
    int linelen = 0;
    int newflen = 0;
    for (int i = 0; newf[i]!='\0'; i++)
    {
        newflen++;
    }
    filePointer = fopen(fname, "r");
    while(fgets(line, bufferLength, filePointer))
    {
        linelen=0;
        for (int i = 0; line[i]!='\0'; i++)
        {
            linelen++;
        }
        char *ptr = strstr(line, newf); //check newf in debugger
        if (ptr != NULL && (linelen==newflen))
        {
            //printf("\nINPUT.txt line is %d characters long and newf is %d
long",linelen,newflen);
            wordExist=1;
            return 0;
        }
    }

    if (wordExist!=1)
    {
        //add word to input file
        FILE * fptr;
        fptr = fopen(fname, "a");
        fputs(newadd, fptr);
        fclose (fptr);
    }
    strcpy(prev,new);
    strcpy(new,"");

    return 1;
    fclose(filePointer);
}

int gameLogic(int newSocket, char *buffer)
{
    strcpy(new, buffer);
    strcpy(newf,"");
    strcpy(newadd,"\n");
    strcat(newf, new);
    strcat(newf,"\n");
    strcat(newadd, newf);
    size_t n = sizeof(prev)/sizeof(char);
    size_t nnew = sizeof(new)/sizeof(char);
    size_t nnewf = sizeof(newf)/sizeof(char);
    char lowernew[101];

    mqd_t dictionary = openMsgQueue("/Dictionary_Check");

    printf("GOT TO THE LOOP\n");
    int disallowed = 0;
    for (int i=0; i<n;i++)
    {
        for (int x = 0; x < nnew && disallowed==0 && new[x]!='\0'; x++) //this
loop checks if there is any disallowed characters in the user entry from this
input file
        {
            for (int y = 0; y < 6 && new[x]!='\0'; y++)
            {
                //printf("\n Iteration %d y iteration %d we're looking at %c in
new and %c in letters\n", x, y, new[x], letters[y]);
                if (new[x]!=letters[y])
                {
                    if (letters[y+1]=='\0')
                        disallowed=1;
                    else
                        continue;
                }
                else
                    break;
            }
        }
        if (disallowed==0)
        {
            if (new[0]==prev[i])
            {
                printf("\nUsed correct characters!\n");
                int j = i;
                int k = 0;
                while ((j<n) && (new[k]==prev[j]) && !((new[k]=='\0') &&
(prev[j]=='\0'))) //checks if the
                {
                    //printf("\n Iteration %d we're looking at %c in new and %c
in prev\n", i, prev[j], new[k]);
                    j++;
                    k++;
                    //printf("The value of j is %d k is %d n is %d", j,k,n);
                }
                if ((j==n) || (prev[j]=='\0') || ((new[k]=='\0') &&
(prev[j]=='\0')))
```

```
                {
                    printf("\nWord is valid!\n");
                    //check if word is a dictionary word
                    printf("\nConverting %s to lower\n",new);
                    for(int w = 0; w<nnewf; w++)
                    {
                        lowernew[w] = tolower(newf[w]);
                    }

                    // Dictionary
                    // int dictionaryCheck(size_t nnewf, char *lowernew, int
newSocket)
                    // FORKING --------------------------------------------------
------------
                    dictionaryCheck(dictionary, nnewf, lowernew, newSocket);

                    // Recieve dictionary check posix message, return 0 if
incorrect
                    if (strcmp(recieveDictionaryMessage(dictionary), "INCORRECT")
== 0)
                    {
                        bzero(buffer, sizeof(buffer));
                        strcpy(buffer, "INCORRECT");
                        printf("INCORRECT DICT\n");
                        minusPlayerScore(added_player, -1);
                        send(newSocket, buffer, 1024, 0);
                        return 0;
                    }

                    wait(NULL);
                    // Used words check
                    //check if word has already been used
```

```
int dup = 0;
for (int j = 0; j < 100; j++)
{
    if(strcmp(new, usedWords[j]) == 0)
    {
        dup = 1;   // got a duplicate
        break;
    }
}
if (dup == 0)
{    // not a duplicate: add it to usedWords
    strcpy(usedWords[noUsedWords+1], new);
    noUsedWords += 1;

    // Send correct message
    printf("CORRECT: %s", buffer);
    bzero(buffer, sizeof(buffer));
    strcpy(buffer, "CORRECT");
    send(newSocket, buffer, 1024, 0);
    strcpy(prev, new);
    return 1;
}
if(dup)
{
    //penalise
    bzero(buffer, sizeof(buffer));
    strcpy(buffer, "INCORRECT");
    printf("INCORRECT DUPLICATE\n");
    minusPlayerScore(added_player, -2);
    send(newSocket, buffer, 1024, 0);
```

```
                return 0;
            }
        }
        else
        {
            bzero(buffer, sizeof(buffer));
            strcpy(buffer, "INCORRECT");
            printf("INCORRECT NOT VALID\n");
            minusPlayerScore(added_player, -1);
            send(newSocket, buffer, 1024, 0);
            return 0;
        }
    }
    if(i==(n-1))
    {
        bzero(buffer, sizeof(buffer));
        strcpy(buffer, "INCORRECT");
        printf("INCORRECT CHARS");
        minusPlayerScore(added_player, -1);
        send(newSocket, buffer, 1024, 0);
        return 0;
    }
}
else
{
    bzero(buffer, sizeof(buffer));
    strcpy(buffer, "INCORRECT");
    printf("INCORRECT DISALLOWED\n");
    minusPlayerScore(added_player, -1);
    send(newSocket, buffer, 1024, 0);
```

```
            return 0;
        }
    }
}
```