
SPACE
SHOOTER

Created by:

Yousuf ahmed Siddiqui (21k-4594)
team leader

Muhammad Aalyan (21k-3432)

Syed arham
(21k-4841)

Taha abbas
(21k-3438)

Our project proposal..

INTRODUCTION

The project that our group is preparing is basically a space shooting game, which might be a little old fashioned and cliche, but our focus is to enhance the game and give it a new face with our fancy innovations and improvements. In this game we will provide real time shooting and real time movement by a spaceship; the spaceship can move sideways and shoot the asteroids/obstacles that are obstructing it's way. Moreover the spaceship will have 3 life's and everytime the spaceship fails to shoot the obstacle and obstacle passes down the ship, it will lose a life. In addition to this, if a spaceship fails to shoot or dodge the obstacle, and the obstacle directly strikes the spaceship, the game will automatically end.

EXISTING PROBLEM

The problem that the existing game comprises of is that the game is not very fancy and does not appear much appealing to the player. Moreover, there is only a single difficult level set for the game and therefore it appears too easy for some players and too difficult for others. Also there is no record for the previous high scores set by the player, and no stats are available.

PROPOSED SOLUTION

The solutions that our group proposes for the further development and improvement of the game is that we will set various diffculty levels for the players i.e : easy/medium/difficult so that the player can choose the difficulty level of their desire, and with different difficulty levels the speed of the incoming obstacles will also vary . In addition to this, to make the game look more fancy we will also provide different skins(colors and designs) of the spaceship. Also the user will be provided with their previous high scores and a record of theirs stats will also be shown.

…We added the difficulty level

and the skins...

higher difficulty comes with lesser lives

enemy swarm is trying to shoot us

if the enemy laser passes by our spaceship life=life-1;

High score can be seen in score menu or at the end of the game

There are 4 total skins

To make the game interesting we increased the difficulty by moving the enemy ships left and right

How to play;

Move your ship by right or left arrow!

Shoot with upwards arrow

Use right arrow key to select an option.

Game ends if (
enemy swarm reaches our home base
||enemy laser destroys our ship
||we have no more lives left
)

Code distribution:

Yousuf ahmed Siddiqui:

- World generation
- Player generation
- Laser trail
- Enemy random laser generation+ moving the trail forwards
- Score incrementing conditions
- Explosion
- difficulty level+ lives
- whole game logic + researcher
- screen transitions
- printing world
- HELPING ALL OTHER TEAMMATES

M. Aalyan:

- loading animation
- filing the highscore
- enemy generation conditions
- contributed by making game fullscreen
- game ending conditions
- player movement
- controls (laser generation and observing the keypresses)
- enemy subtle movement

Syed arhum:

- designer
- designing title logo
- designing gamover and congratulations logo
- made instructions menu
- frontend and backend coding of skins
- added sounds to the game

taha abbas:

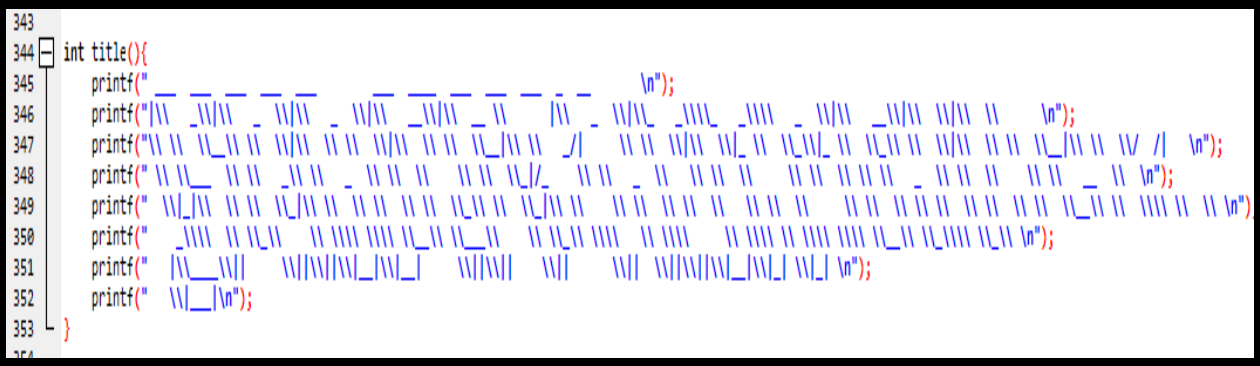
- jump function
- slow functions
- locatex(), locatey()
- adding arrows to animate the game

- helped arhum into developing the skin prototype

Moving onwards to Code description;

SYED ARHAM AHMED – 21K-4841

TASK 1 :- Printing Title



```
int title(){
    printf("  __ __ __ __ __      __ ___ __ __ __ _ _                \n");
    printf(" /\\ _\\/\\ _ \\/\\ _ \\/\\ _\\/\\ _ \\   /\\ _ \\/\\/\\_ _\\\\_ _\\\\ _ \\/\\/\\ _\\/\\/\\ \\    \n");
    printf("\\ \\ \\_\\ \\ \\/\\ \\ \\ \\/\\ \\ \\ \\_/\\ \\  _/|   \\ \\ \\/\\ \\/_ \\ \\_\\/_ \\ \\_\\ \\ \\/\\ \\ \\ \\_/\\ \\ \\/ /|  \n");
    printf(" \\ \\ \\__ \\ \\  _\\ \\  _ \\ \\ \\  \\ \\ \\_/_   \\ \\  _ \\  \\ \\ \\  \\    \\ \\ \\ \\ \\ \\  _ \\ \\ \\  \\ \\  _ \\ \\ \n");
    printf("  \\/_/\\ \\ \\ \\ \\_/\\ \\ \\ \\ \\ \\ \\_\\ \\ \\_/\\ \\   \\ \\ \\ \\ \\ \\ \\  \\ \\ \\    \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\_\\ \\ \\\\\ \\  \\ \\n");
    printf("    _\\\\  \\ \\_\\   \\ \\\\\ \\\\\ \\_\\ \\__\\   \\ \\_\\ \\\\\   \\ \\\\\    \\ \\\\\ \\ \\\\\ \\\\\ \\_\\ \\_\\\\\ \\_\\ \n");
    printf("   /\\___\\/|    \\/|/\\\\/|\\/_/\\\\/_|    \\\\/|\\\\/|    \\\\/|     \\\\/| \\\\/|\\\\/|\\\\/_/\\\\/_| \\\\/_| \n");
    printf("   \\\\/___|\n");
}
```

- First I declared a title function and did the rest of work under it.
- Then I designed the title (SPACE SHOOTER) on a text document and copied it here line by line using printf.
- Also when I used a single \ an error appeared, therefore I had to use double \\ . This is because when we use single \ , the compiler expects us to write a variable after it to complete the sequence like \n or \a .

TASK 2 :- Printing “CONGRATS” with beep and delay

```
127
128 int congrats(){
129     //c
130     Beep(200,50);
131     printf("       __    \n");
132     printf("  __|\\   \\  \n");
133     printf(" /   /\\   \\ \n");
134     printf("|    |  |    |\n");
135     printf("|    |  |_|\n");
136     printf("|    |    _ \n");
137     printf("|    |  |    |\n");
138     printf("|\\___\\/   /|\n");
139     printf("| |  /_/ |\n");
140     printf(" \\|__|    | /\n");
141     printf("  \\( |_|/ \n");
142     printf("     '   )/    \n");
143     printf("         '     \n");
144
145     slow(3);
146     //o
147     Beep(200,50);
148     jump(13,0);
149     printf("       __    \n");
150     jump(13,1);
```

```
145     slow(3);
146     //o
147     Beep(200,50);
148     jump(13,0);
149     printf("       __    \n");
150     jump(13,1);
151     printf("   __|\\   \\   \n");
152     jump(13,2);
153     printf("  /   /\\   \\  \n");
154     jump(13,3);
155     printf(" /   /  \\   \\ \n");
156     jump(13,4);
157     printf("|    |    |    |\n");
158     jump(13,5);
159     printf("|    |    |    |\n");
160     jump(13,6);
161     printf("|\\   \\  /   /|\n");
162     jump(13,7);
163     printf("| \\__\\/__/ |\n");
164     jump(13,8);
165     printf(" \\ |  ||    | /\n");
166     jump(13,9);
167     printf("  \\|_||__|/ \n");
168     jump(13,10);
169     printf("    \\(    )/   \n");
170     jump(13,11);
171     printf("     '    '    \n");
```

- For this task various functions were used as follows:-

1. Slow; it is a user defined function, and its operation is to cause a delay after printing each letter.
2. Beep; it is a pre-defined function in <windows.h> and its purpose is to cause a beep sound while printing each letter. Also we can control the frequency of sound and its duration.
3. Jump; it is also a user defined function, its purpose is to switch/change the axis(x,y) before printing each letter…….

- For this task I first printed the letter C on (0,0) axis and also added a beep sound to it.
- After this,before printing every other letter i used slow function to cause a short delay before it starts printing the other letter and also added beep sound to every letter.
- Moreover jump function was also used continuously to change the axis of printing according to my need, or else it would change

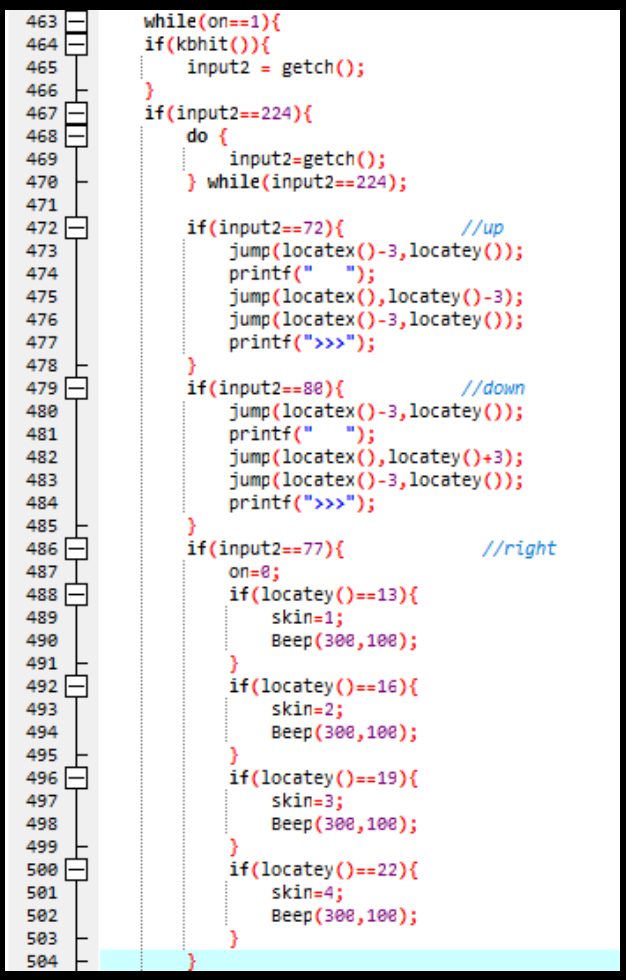
the line and start printing from the extreme left which I did not want.

TASK 3 :- Adding Instructions to the game

```
556
557 int instructions(){
558     jump(1,1);
559     printf("<<<");
560     jump(10,5);
561     printf(" _____");
562     jump(10,6);
563     printf("|welcome to instructions!                                                  |\n");
564     jump(10,7);
565     printf("|-------------------------------------------------------------------------|\n");
566     jump(10,8);
567     printf("|1 ~ player can choose out of 4 skins once the game starts                 |\n");
568     jump(10,9);
569     printf("|2 ~ player can choose difficulty level once the game starts               |\n");
570     jump(10,10);
571     printf("|3 ~ OBJECTIVE is to finish enemy swarm before it hits our spaceship       |\n");
572     jump(10,11);
573     printf("|4 ~ if an enemy laser passes by you ,your life is decreased by one level |\n");
574     jump(10,12);
575     printf("|5 ~ number of lives of a spaceship depends on the difficulty level        |\n");
576     jump(10,13);
577     printf("|6 ~ if player shoots the enemy laser score increases by 100               |\n");
578     jump(10,14);
579     printf("|7 ~ if player shoots enemy spaceship score increases by 50                |\n");
580     jump(10,15);
581     printf("|8 ~ player wins if total enemies left are zero                            |\n");
582     jump(10,16);
583     printf("|_____|");
584     jump(60,30);
585     printf("MADE BY:");
586     jump(60,32);
587     printf("yousuf ahmed siddiqui K21~4594");
588     jump(60,33);
589     printf("syed arhum ");
590     jump(60,34);
591     printf("taha abbas ");
592     jump(60,35);
593     printf("muhammad aliyan");
594 }
```

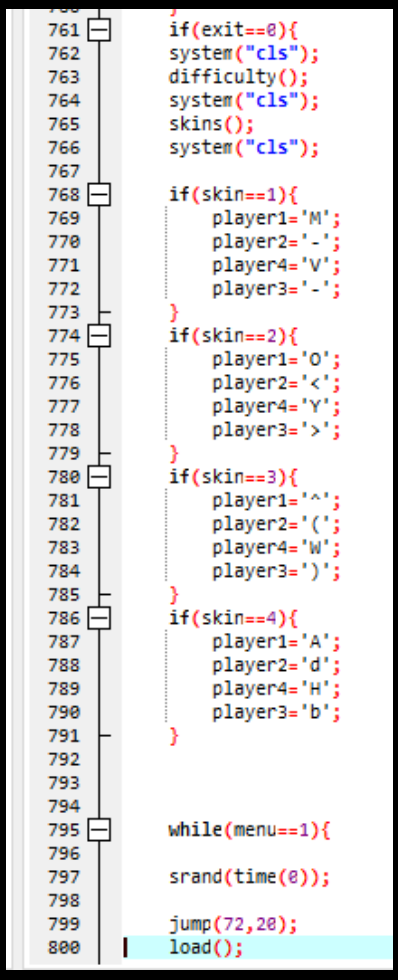
- For this task I just simply made an instructions function and did my working inside the function.
- Jump function was also constantly used in this task while printing, in order to change the axis of printing accordingly.
- The function holds instructions regarding the game (how to play) and the options available. Also the names of the developers is also added.

TASK 4 :- Skins (Front and back end coding)



```
while(on==1){
if(kbhit()){
    input2 = getch();
}
if(input2==224){
    do {
        input2=getch();
    } while(input2==224);

    if(input2==72){          //up
        jump(locatex()-3,locatey());
        printf("   ");
        jump(locatex(),locatey()-3);
        jump(locatex()-3,locatey());
        printf(">>>");
    }
    if(input2==80){          //down
        jump(locatex()-3,locatey());
        printf("   ");
        jump(locatex(),locatey()+3);
        jump(locatex()-3,locatey());
        printf(">>>");
    }
    if(input2==77){          //right
        on=0;
        if(locatey()==13){
            skin=1;
            Beep(300,100);
        }
        if(locatey()==16){
            skin=2;
            Beep(300,100);
        }
        if(locatey()==19){
            skin=3;
            Beep(300,100);
        }
        if(locatey()==22){
            skin=4;
            Beep(300,100);
        }
    }
```



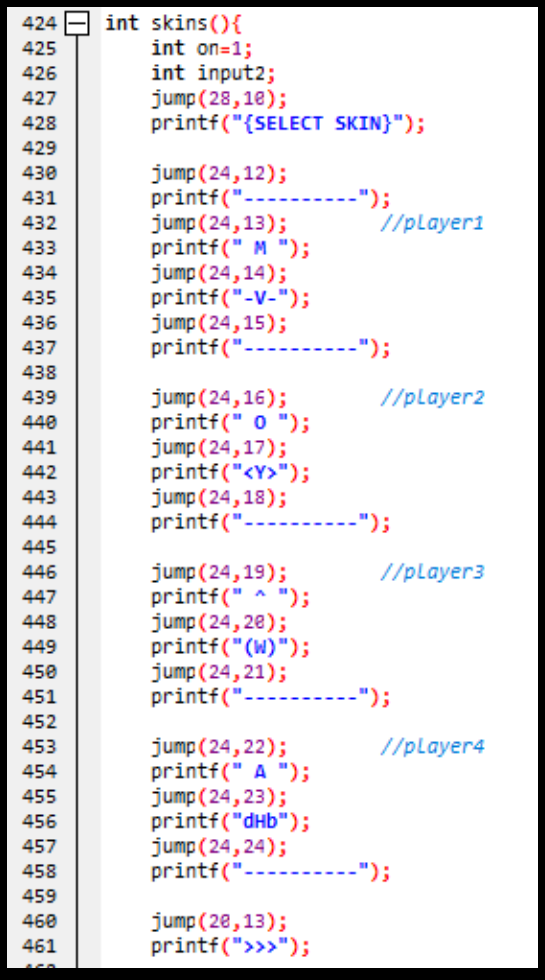
```
    if(exit==0){
    system("cls");
    difficulty();
    system("cls");
    skins();
    system("cls");

    if(skin==1){
        player1='M';
        player2='-';
        player4='V';
        player3='-';
    }
    if(skin==2){
        player1='O';
        player2='<';
        player4='Y';
        player3='>';
    }
    if(skin==3){
        player1='^';
        player2='(';
        player4='W';
        player3=')';
    }
    if(skin==4){
        player1='A';
        player2='d';
        player4='H';
        player3='b';
    }


    while(menu==1){

    srand(time(0));

    jump(72,20);
    load();
```



```
int skins(){
    int on=1;
    int input2;
    jump(28,10);
    printf("{SELECT SKIN}");

    jump(24,12);
    printf("----------");
    jump(24,13);          //player1
    printf(" M ");
    jump(24,14);
    printf("-V-");
    jump(24,15);
    printf("----------");

    jump(24,16);          //player2
    printf(" O ");
    jump(24,17);
    printf("<Y>");
    jump(24,18);
    printf("----------");

    jump(24,19);          //player3
    printf(" ^ ");
    jump(24,20);
    printf("(W)");
    jump(24,21);
    printf("----------");

    jump(24,22);          //player4
    printf(" A ");
    jump(24,23);
    printf("dHb");
    jump(24,24);
    printf("----------");

    jump(20,13);
    printf(">>>");
```

Front end coding:

I designed the most suitable skin patterns on notepad++. Which were used in the skin menu template

Backend coding:

A player is divided in 4 characters

All 4 player elements since visible in world array,

carry ability to be customized

if user selects 1st preference, the corresponding characters are then assigned to all player elements

Mr Aalyan (3432)

```
//enemy generation in world at start
        for(y=0;y<sizey;y++){
                for(x=0;x<sizex;x++){
                        if( (y%2==0) && y<(sizey/3) && (x>3) && (x<=sizex-4) && (x%2==0) ){
                                world[y][x]=enemy;
                                totenemy++;
                        }
                        else {
                                world[y][x]= ' ';
                        }
                }
        }
```

Enemy Is generated here :

- **Sizey** ➔is the total y console on which game is played
- **Sizex** ➔is the total x console on which game is played
- **y%2** ➔is done so that enemy is generated on each alternate y coordinate of the console
- **sizey/3** ➔is done so that enemy is generated on the one third of the enire console
- **(x>3) && (x<=sizex-4) && (x%2==0)**➔ is done so that on each alternate x coordinate enemy is generated. here **x>3** is done so that enemy is generated not on the borders of the console but on x coordinate that is greater than 3 and **x<=sizex-4** is done so that enemy is generated not at the right borders of the console but it should be 4 coordinate less than the border and **x%2==0** is done so that enemy is generated on the alternate x coordinate of the console .

```
//player skin generation and spawning place
```

```
world[sizey-2][sizex/2]=player1;

world[sizey-1][(sizex/2)-1]=player2;

world[sizey-1][(sizex/2)+1]=player3;

world[sizey-1][sizex/2]=player4;
```

- **world[sizey-2][sizex/2]=player1➔** is done so that player 1 is spawn on 2 coordinate less than the entire y console **(sizey-2)** on the exact middle of the x console (**sizex/2)**
- **world[sizey-1][sizex/2] =player4➔** is done so that player 4 is spawn on 2 coordinate less than the entire y console **(sizey-1)** on the exact middle of the x console (**sizex/2)**
- **world[sizey-1][(sizex/2)-1]=player2➔** is done so that player 2 is spawn on 1 coordinate less than the entire y console **(sizey-1)** on the exact middle of the x console minus 1 x coordinate so that a new skin is made **((sizex/2)-1)**
- **world[sizey-1][(sizex/2)+1]=player3➔** is done so that player 3 is spawn on 1 coordinate less than the entire y console **(sizey-1)** on the exact middle of the x console minus 1 x coordinate so that a new skin is made **((sizex/2)+1)**

```
//controls


If (kbhit()) {
 input = getch();
     }
              if(input==224){
              do {
            input=getch();
       } while(input==224);


                          if(input==77){              //right
                                              for(x=0;x<sizex;x++){
                                                          if(world[sizey-2][x-2]==player1){//bahar nikal jaye
                                                                      world[sizey-2][x-1]=player1;
                                                                      world[sizey-2][x-2]=' ';


                                                                      world[sizey-1][x-1]=player4;
                                                                      world[sizey-1][x]=player3;
                                                                      world[sizey-1][x-2]=player2;
                                                                      world[sizey-1][x-3]=' ';



                                                                      break;
                                                          }
                                              }
                                    }
                          if(input==75){            //left
                                              for(x=0;x<sizex;x++){
                                                          if(world[sizey-2][x+2]==player1){
                                                                      world[sizey-2][x+2]=' ';
```

```
                                        world[sizey-2][x+1]=player1;


                                        world[sizey-1][x]=player2;
                                        world[sizey-1][x+1]=player4;
                                        world[sizey-1][x+2]=player3;
                                        world[sizey-1][x+3]=' ';



                                        break;
                                    }
                                }
                        }
                        if(input==72 && mod2>1){
                                lasready=1;
                                Beep(400,30);
                                mod2=0;
                        }
        }
        system("cls");
```

<<<<<<<77 asc11 value is for the right arrow key and 75 asc11 is for the left arrow key>>>>>>>>

- input = getch()➔ this function would get the command of the arrow we would like to shift
- input==77➔ if right arrow key is pressd then our player would shift to the right (77 is the asc11 value) of arrow key right.
- input==75➔ if left arrow key is pressd then our player would shift to the left (75 is the asc11 value) of arrow key left.
- for(x=0;x<sizex;x++)➔ this would insure that our player moves in x direction till our defined sizex and would increment till
- world[sizey-2][x-1]=player1➔player 1 place is changed hence it would be shifted right (world[sizey-2][x-2]=' ') and this would ensure that space would be printed on the current place of player 1 .
- world[sizey-1][x-1]=player4➔ player 4 place is changed hence it would be shifted right (world[sizey-1][x-3]=' ') and this would ensure that space would be printed on the current place of player 4
- world[sizey-1][x-1]=player3➔ player 3 place is changed hence it would be shifted right (world[sizey-1][x-3]=' ') and this would ensure that space would be printed on the current place of player 3

- world[sizey-1][x-1]=player2➔ player 2 place is changed hence it would be shifted right (world[sizey-1][x-3]=' ') and this would ensure that space would be printed on the current place of player 2
- for player 1 we use size[y-2]and for player 2,3,4 we use size[y-1], because player 1 is on the top of the other three player 2,3,4.
- **If (kbhit())**➔ this means if key board key is pressed

JUMP FUNCTION THIS SIMPLY MOVES THE CURSOR To THE ALLOCATED LOCATION WE WANT BASIC FUNCTION IS JUMP(X,Y) WHERE X AND Y ARE THE COORDINATES WE WANT OUR CURSOR TO BE POINTED .

System("MODE 1000,1000") THIS FUNCTION IS USED SO THAT OUR CMD PANEL WOULD AUTOMATICALLY BE IN THE FULLSCREEN ZONE .

```
#include <stdio.h>
#include <stdlib.h>
#include <windows.h>

void readFile(FILE * fPtr){
   char ch;
   do {
      ch = fgetc(fPtr);
      putchar(ch);
   }while(ch != EOF);
}

void check(FILE * fPtr){
   int ch,i,n=0,gre=-1;
   int highscore[100]={0};
   for(i=0;i<100000;i++){
//       fscanf(fPtr,"%d",&ch);
//       fscanf(fPtr,"\n");
                        if(fscanf(fPtr,"%d",&ch)==EOF){
       Beep(200,200);
                        break;
                        }
```

```
        printf("\n\t%d ",ch);
                highscore[i]=ch;
                        n++;
                }

                for(i=0;i<n;i++){
                        if(highscore[i]>gre){
                                gre=highscore[i];
                        }
                }

                printf("\n\n\n\n\n\nHIGHSCORE:%d",gre);
}

int main(){
    FILE *fPtr;
    FILE *f;
    char name[50];
    int score;



    fPtr = fopen("HIGHSCORE.txt", "a");
    f=fopen("sc.txt","a");

    if (fPtr == NULL || f==NULL ){
        printf("\nUnable to open ");
    }

    printf("\nEnter your name: ");
    gets(name);
    printf("\nEnter your score: ");
    scanf("%d",&score);

    fprintf(f,"%d\n",score);

    fprintf(fPtr,"\n");
```

```
    fputs(name, fPtr);
            fprintf(fPtr,"\t%d",score);

    fPtr = freopen("HIGHSCORE.txt", "r", fPtr);
       f = freopen("sc.txt", "r", f);
    printf("\nSuccessfully appended data to file. \n");
    printf("Changed file contents:\n\n");
    readFile(fPtr);
    check(f);
    fclose(fPtr);
    fclose(f);
}
```

- **(FILE * fPtr)➔** Simply the declaration of file pointer
- **int highscore[100]={0}➔** highscore array is defined here and only 100 elements score would be made .
- **FPRINTF** is used to write on the file.
- Firstly HIGHSCORE named file is opened to append and is checked that if there is a file named highscore then would be present and if the file is not found named highscore then error would be printed. Name(gets(name)) and score(scanf("%d",&score)) is taken and printed on the file . fprintf is used to print score on the new file with name score then highscore file is reopened and is reopened to read and score file is also reopened to read and then read function and check function is called and then both the files are closed .
- **void readFile(FILE * fPtr)➔** main function of this read function is that it only reads from the file jit would read till the end of file .
- **void check(FILE * fPtr)➔** all the score is stored in a file . ch mein score save horha hai .
- **for(i=0;i<100000;i++)➔** name aur score highscore k array mein store horha hai aur is loop mein n++ horha hai and n is number of games played.
- **for(i=0;i<n;i++)➔** is loop mein ham array mein compare karwa rhay hai and then printing the highscore.

```
int load(){
        int i,ld=0,cnt=0;
        int
ch[36]={177,176,176,176,176,176,176,176,176,176,176,176,176,176,176,176,176,176,176,176,176,176,176,176,176,176,176,176,176,176,176,176,1
76,176,176,176};
        system("cls");
        jump(72,20);
        printf("LOADING..");
        while(cnt<3){
        if(ld==60){
                ld=0;
                cnt++;
        }
        jump(58,23);
        for(i=0;i<36;i++){
                printf("%c",ch[i]);
        }
        if(ld<=5){
                ch[0]=177;
        }
        else if(ld<=14){
                ch[0]=178;
        }
        else if(ld<=20){
                ch[0]=177;
        }
        else if(ld>20 ){
                ch[0]=176;
        }
        for(i=35;i>=0;i--){
                ch[i+1]=ch[i];
        }
        slow2(2);
        jump(58,23);
        printf("                                    ");
        jump(58,23);
        ld++;
 }
```

```
system("cls");    }
```

- **176,177,178➔** is used for the intensity of colors; 176 is for the lightest and 178 is for the darkest color.
- **for(i=0;i<36;i++)➔** in this loop all 36 times lght is printed that is 176 pre defined.

```
for (y=0;y<sizey;y++){

        if (world[y][0]==enemy){// if enemy touches left border ,now the upcoming direction is to be to the right and also generates signal
to drop down the array

            direction=1;

            down=1;

            break;

        }

           if (world[y][sizex-1] == enemy){ // if enemy touches right border ,now the upcoming direction is to be to the left and also
           generates signal to drop down the array
                       direction=0;
                       down=1;
                       break;
           if( direction==0){
                                 if(c%10==0){
                                  for(x=0;x<sizex;x++){
                                 for(y=0;y<sizey;y++){
                                                       if(world[y][x]==enemy &&down==0){
                                                       world[y][x]=' ';
                                                       world[y][x-1]=enemy;
                                                       }
                                                       if(world[y][x]==enemy &&down==1){
                                                       world[y][x]=' ';
                                                       world[y+1][x-1]=enemy;
                                                       }
                                                 }
                                 }
                                 }
                       }
                   else{
                          if(c%10==0){
                                     for(x=sizex-1;x>=0;x--){
                                     for(y=0;y<sizey;y++){
                                                       if(world[y][x]==enemy &&down==0){
                                                       world[y][x]=' ';
                                                       world[y][x+1]=enemy;
                                                       }
                                                       if(world[y][x]==enemy &&down==1){
                                                       world[y][x]=' ';
                                                       world[y+1][x+1]=enemy;
                                                       }
                                                 }
```

```
        }
        }
}
```

- **if (world[y][0]==enemy)➔** if enemy is on the left border then it have to move down towards the right hand side
- **if (world[y][sizex-1]==enemy)➔** if enemy is on the right border than it have to travel down and towards the left hand side
- **direction=0 & direction=1 ➔** Is towards left and direction 1 is towards right
- **down=1➔** shift down
- **if(c%10==0)➔** moves on every 10 iteration of the while loop

END GAME CONDITIONS

```
for(x=0;x<sizex;x++){
       if(world[sizey-2][x]==enemy){
           gameover=1;
       }
   }
```

- **if(world[sizey-2][x]==enemy)➔** end game condition

Taha abbas(3438)

Locate x locate y: -

These two functions are used as a help in the shape of coordinates. I have taken these 2 functions in which the width of the game has been assigned. For example I use these two functions to move the arrow(›››)

```
int locatex(){
    CONSOLE_SCREEN_BUFFER_INFO csbi;
    GetConsoleScreenBufferInfo(GetStdHandle(STD_OUTPUT_HANDLE), &csbi);
    return csbi.dwCursorPosition.X;
}

int locatey(){
    CONSOLE_SCREEN_BUFFER_INFO csbi;
    GetConsoleScreenBufferInfo(GetStdHandle(STD_OUTPUT_HANDLE), &csbi);
    return csbi.dwCursorPosition.Y;
}
```

the game is mostly based upon these two function as to move the cursor to the left or the right. To sum up, this is the coordinates of the game.

Slow: -

This function is used as a function of time. Its datatype is integer and it is used in the code to stall time for a function

```
void slow(int sec){
    int millisec=500*sec;
    clock_t start_time =clock();
    while(clock()<start_time+millisec);}
```

Jump:

This function is used with the locate x and locate y function. It is used to jump from a coordinate to another. It is used in the place of a function which is named as goto function (It is a predefined function). Goto function is not used as it alters the sequential flow of logic that is the characteristic of C language.

```
void jump(int x, int y){
    COORD coord;
    coord.X =x;
    coord.Y =y;
    SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE), coord);
}
```

Arrow in the code (›››) :-

The arrow indicates the cursor in the game. It is used to set the cursor on the option that we have to select. Using the co-ordinates, we assign the value to the arrow in the code and it is selected in every option of the game whether it is an instruction menu or to select the difficulty level of the game. If we decrease/increase one of the coordinate in the code the arrow goes to the position and set it there. The arrows in the code works through the coordinates as we can alter the value in every aspect. It also works with the jump function for example,

```
int difficulty(){
      int on=1;
      int input2;
      jump(24,10);
      printf("{SELECT DIFFICULTY}");
      jump(24,13);
      printf("easy");
      jump(24,15);
      printf("medium");
      jump(24,17);
      printf("hard");
      jump(25,13);
      printf(">>>");
```