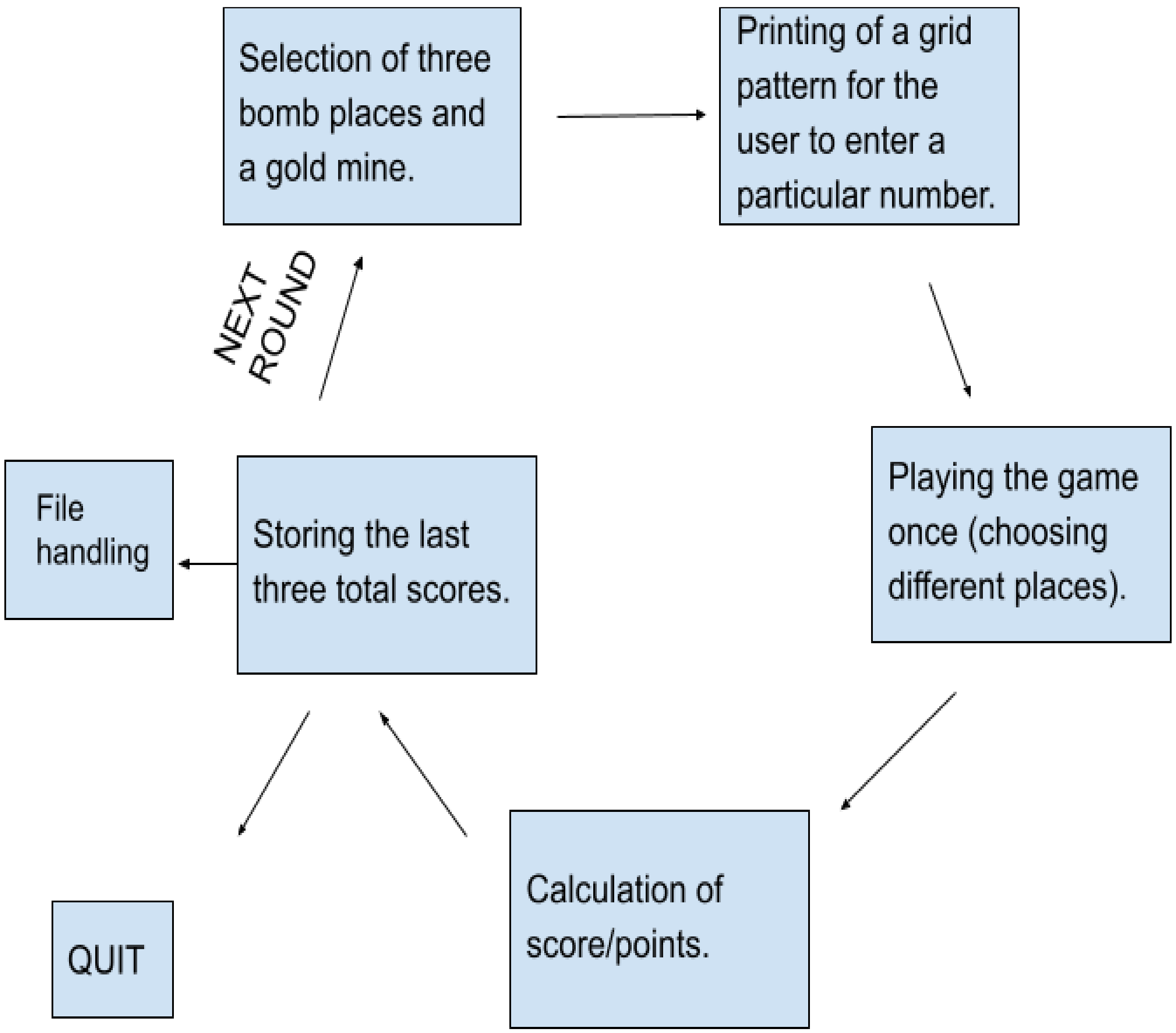
TREASURE HUNT

Problem Statement:

(a) Consider a grid of 4x5.
(b) Use the random function to position the 3 bombs in the grid every time the game is started,
(c) Use the random function to position 1 box of gold coins in the grid every time the game is started.
(d) Use the scanf function to take player input.
(e) If the number entered by the user = number where the bomb is positioned then the player loses and the game ends.
(f) If the number entered by the user = number where the box of gold coins is positioned then the player wins and gets 100 points. A new game can be started with the player score retained.
(g) If the number entered by the user != number where the bomb is positioned or the box of gold coins is positioned then the player must input a new number and get 5 points as a bomb is not encountered. After 5 chances, the game must be restarted but the player score must be retained.
(h) When the player wishes to terminate the game, the final score must be shown.

Steps:


Selection of three bomb places and a gold mine.
Printing of a grid pattern for the user to enter a particular number.
Playing the game once (choosing different places).
Calculation of score/points.
Storing the last three total scores.
File handling
QUIT
NEXT ROUND

Implementation:

- To generate random numbers for placing the bombs and the gold box using a function named “rand_num_gen()”. (We have used the inbuilt functions “rand()” and “srand()”.
- We used a nested ‘for’ loop to print a pattern which resembles a grid of 4*5 dimensions.
- We defined a function named “game()”, in which we input numbers (maximum five numbers can be input) from the user, grid numbers, which points to the chosen box.
- Conditional loops are used to show error in case the input number is more than 20 or if a previous input is given again.
- We used if-else to compare the chosen box for position of gold coin or bombs and update score(points) accordingly.
- An array is updated to store the score after each game round to store the scores of previous individual rounds. After the game ends, scores from the previous 3 rounds are printed on the screen.
- In case of input landing on a mine/bomb, a new game is started.
- After 5 consecutive rounds, a new game option appears. Score is retained.

Additional features:

1. Grid display- A rectangular grid with numbered boxes is displayed to the user in each game.
   At the end of the round, the position of the bomb or gold coin encountered is also represented on the grid.
2. At the end of each round, the user is given options to either continue the game another time or exit.
3. We are taking the feedback about the game from all the users in the form of rating.
4. Taking names from users and storing their game score and name details in structure.
5. Use of File Handling and structures to store user name and display previous 3 games score.
6. We did not provide any supporting file along with our code so people can play the game by just downloading the finalproject.c file without any additional files.
7. Displaying smileys after every round based on the result.
8. If the user scores more than 100 points in three consecutive games we are giving him 3 diamonds which he can use to exchange coins where each diamond is exchanged for 50 coins.
9. We are updating the number of diamonds the user has similar to the score part and we are also printing the number of diamonds of the particular user in each game.

VARIABLES:

Variable name	Data Type	Application
bomb1, bomb2, bomb3	int	Stores the location of respective bombs.
gold	int	Stores the location of gold mine
data_format	struct	stores the format in which data is stored in the file
title	char array	stores the title which is printed before printing scores of a particular game.
score	int	stores score in the structure data_format
name	char array	Stores name of the user in struct data_format.
num_diamonds	int	Stores number of diamonds present with the user .
score_det	structure array	array of type data_format.
rand_num_gen	function	generates four random numbers between 1 to 20.
spl_var	int	Stores the location of the cell on which the user is going to land.
ch	char	Used as a parameter for the pattern funct. To print different grids and smileys.
count	int	Used to go with the progression of the grid
enter_val_check	Int array	Used to store user’s input in the game.
points	int	stores score of the user in that game.

i, j	int	for Loop variables
username	Char array	stores the name of the user.
choice	char	stores the user choice to play again or Quit
curr_score	int	stores points
fp	File pointer	used in file handling operations
user[50]	Structure array	used to store all details regarding user
rec_size	int	size of repeating records in the file
num_rec_presnt	int	number of records already present in the file
user_present	bool	Used to know whether user already played a game or not
score_change	char	takes the user choice to use diamonds or not
user_num_diamonds	int	number of diamonds user wants to exchange.
rating	int	Stores rating given by the user

FUNCTIONS:

Function name	**Return Type**	**Application**
print()	void	To give introductory printing statements for the user to understand the game.
rate()	void	Taking feedback from the player about how our game was.
smiley(char ch)	void	Displays smiley based on result
rand_num_gen()	void	Generates four random numbers for positioning bombs and gold.
pattern(int spl_var,char ch)	void	Prints grid pattern (4*5 grid with

		numbers, marked at certain position if required)
game()	int	Runs the game and returns us the points of that particular game.
main()	int	Controls program flow

STRUCTURES:

structure name	**content**	**Data Types**	**application**
data_format	● Title[50] ● score	char int	Stores the data format in which scores are represented in the game.
user	● Name[50] ● Score_det[3] ● num_diamonds	char data_format int	Stores information about the user.

INBUILT FUNCTIONS:

Function name	**Application**
srand()	If it isn't used then the rand() function would generate the same sequence of random numbers.
rand()	Used to generate random numbers in a range.
fread()	Used to read data in specified number of bytes from a file stream

fwrite()	Used to write data in specified number of bytes into a file stream

Code and explanation:

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <stdbool.h>
#include <string.h>

int bomb1, bomb2, bomb3, gold; //storing positions of bombs and gold coin
```

We declared the variables bomb1,bomb2,bomb3 and gold globally because we are using them in all functions and these values need to be updated along the code. We used time.h header file for passing parameter time(0) for srand function , we used stdlib.h header file for rand function and standard stdio.h header file taking inputs, printing outputs ,etc .

```
struct data_format
{
    char title[50];
    int score;
};
struct user
{
    char name[50];
    struct data_format score_det[3];
    int num_diamonds;
};
```

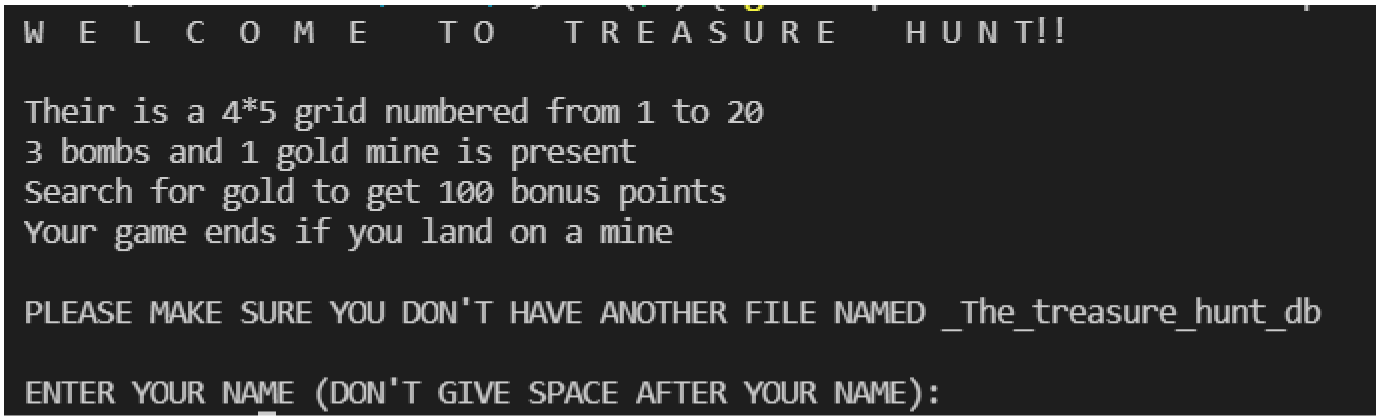
We have used a structure concept to store the name and score of each individual player.

print():

This function gives a basic introduction to the player about the game. Mainly consists of printing statements.

```c
void print()
{
    printf("W  E  L  C  O  M  E    T O T R E A S U R E    H U N T!!\n\n");
    printf("Their is a 4*5 grid numbered from 1 to 20\n");
    printf("3 bombs and 1 gold mine is present\n");
    printf("Search for gold to get 100 bonus points\n");
    printf("Your game ends if you land on a mine\n");
    printf("\n");
    printf("PLEASE MAKE SURE YOU DON'T HAVE ANOTHER FILE NAMED _The_treasure_hunt_db \n");
    printf("\n");
    printf("ENTER YOUR NAME (DON'T GIVE SPACE AFTER YOUR NAME): ");
}
```

- **These statements include : Information about game aspects and instructions about how the game is played.**
- **The user is then asked to input their name.**

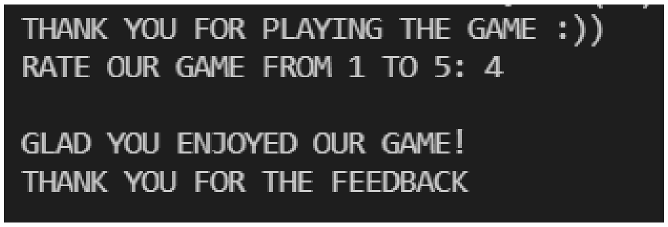


```
W E L C O M E   T O   T R E A S U R E   H U N T!!

Their is a 4*5 grid numbered from 1 to 20
3 bombs and 1 gold mine is present
Search for gold to get 100 bonus points
Your game ends if you land on a mine

PLEASE MAKE SURE YOU DON'T HAVE ANOTHER FILE NAMED _The_treasure_hunt_db

ENTER YOUR NAME (DON'T GIVE SPACE AFTER YOUR NAME):
```

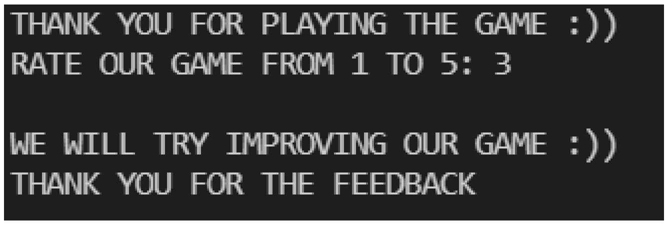
rate():

This function collects the feedback from the player in the form of a rating between 1-5.

```
void rate()
{
    printf("THANK YOU FOR PLAYING THE GAME :))\n");
    int rating;
    printf("RATE OUR GAME FROM 1 TO 5: ");
    scanf("%d",&rating);
    printf("\n");
    if(rating <=3)
        printf("WE WILL TRY IMPROVING OUR GAME :))\n");
    else
        printf("GLAD YOU ENJOYED OUR GAME!\n");
       printf("THANK YOU FOR THE FEEDBACK\n\n");
}
```
	➢ **These statements include : After the user finishes playing, they are asked to rate the game between 1 to 5.**
	
	➢ **A respective default thank you message is then displayed**



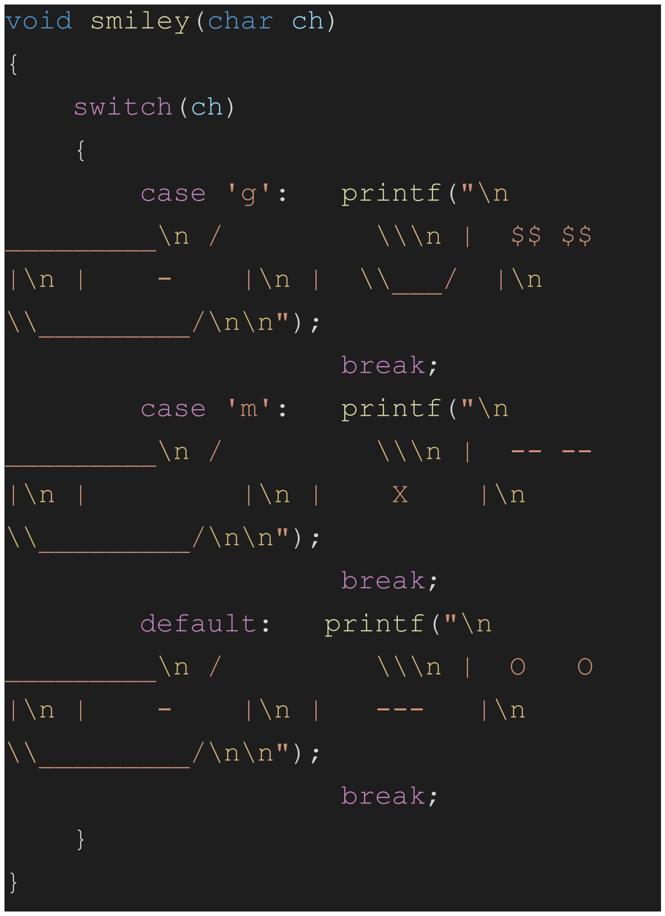
```
THANK YOU FOR PLAYING THE GAME :))
RATE OUR GAME FROM 1 TO 5: 4

GLAD YOU ENJOYED OUR GAME!
THANK YOU FOR THE FEEDBACK
```



```
THANK YOU FOR PLAYING THE GAME :))
RATE OUR GAME FROM 1 TO 5: 3

WE WILL TRY IMPROVING OUR GAME :))
THANK YOU FOR THE FEEDBACK
```

smiley(char ch):

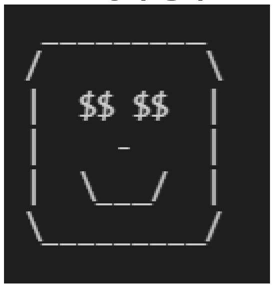
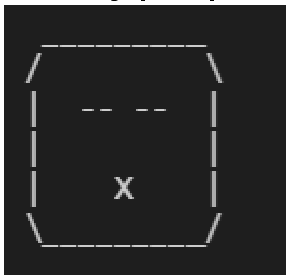
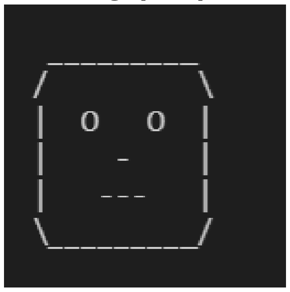
This function shows emoticons/expressions according to “ch” that is used to check whether the person landed on the mine/treasure or the round got over.



```
void smiley(char ch)
{
    switch(ch)
    {
        case 'g':   printf("\n
_________\n /         \\\n |  $$ $$
|\n |    -    |\n |  \\___/  |\n
\\_________/\n\n");
                    break;
        case 'm':   printf("\n
_________\n /         \\\n |  -- --
|\n |         |\n |    X    |\n
\\_________/\n\n");
                    break;
        default:   printf("\n
_________\n /         \\\n |  O   O
|\n |    -    |\n |   ---   |\n
\\_________/\n\n");
                    break;
    }
}
```

- ➢ **Using different print statements, we print different expressions.**
- ➢ **3 types of smileys are constructed (happy, sad and normal).**
- ➢ **The pattern function has one parameter (ch), which is used to differentiate the 3 types of grids using switch case.**
- ➢ **We have used \n , \ , / , O, $ and X are used to create the emoticon.**
- ➢ **Sequence of printing: head , eyes, nose and then mouth.**

[redacted]	(Top to Bottom)

smiley('g')	smiley('m')	smiley('o')
		

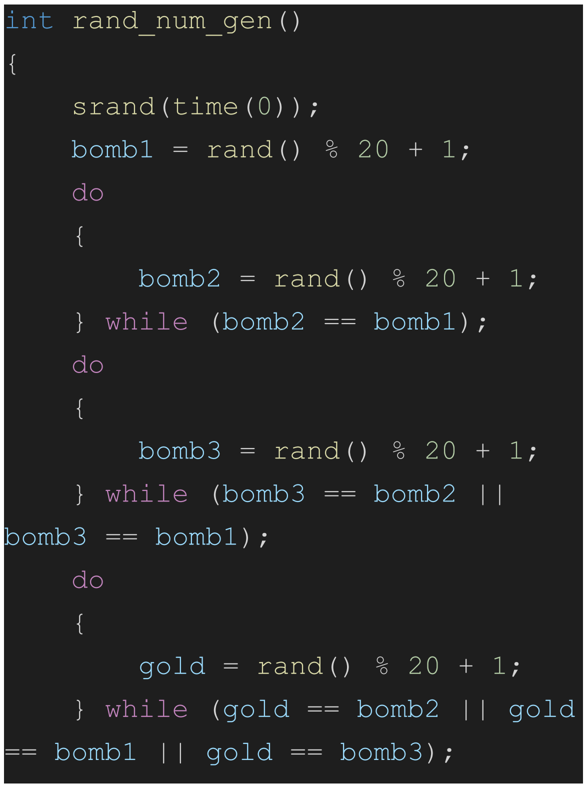
```
 __________
/          \
|  $$ $$   |
|    -     |
|  \___/   |
\__________/
```

```
 __________
/          \
|  -- --   |
|          |
|    x     |
\__________/
```

```
 __________
/          \
|  o   o   |
|    -     |
|   ---    |
\__________/
```

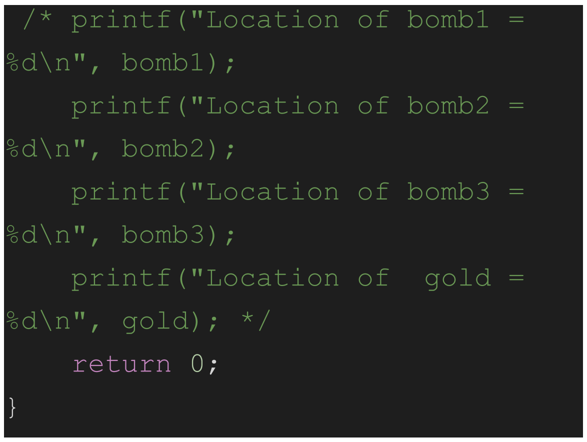
rand_num_gen() :

This function generates four random and unequal values(with complete uncertainty) to be stored in the variables “bomb1”,“bomb2”,“bomb3” and “gold” that show position of respective things. Inside this function we used rand() and srand() built in functions of the library stdlib.h and time.h.

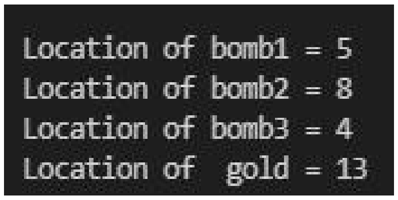


```
int rand_num_gen()
{
    srand(time(0));
    bomb1 = rand() % 20 + 1;
    do
    {
        bomb2 = rand() % 20 + 1;
    } while (bomb2 == bomb1);
    do
    {
        bomb3 = rand() % 20 + 1;
    } while (bomb3 == bomb2 ||
bomb3 == bomb1);
    do
    {
        gold = rand() % 20 + 1;
    } while (gold == bomb2 || gold
== bomb1 || gold == bomb3);
```

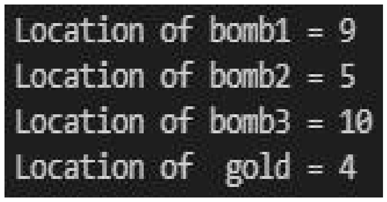
- ➢ rand() generates random numbers.
- ➢ srand() makes sure that the random numbers generated won't be the same for every game .
- ➢ bomb1 takes a random value(1 to 20) i.e. `rand() % 20` gives value from 0 to 19.
- ➢ bomb2 takes the value ensuring it is not equal to bomb1.
- ➢ Similarly, bomb3 and gold are given some random values.
- ➢ do-while loop is used to ensure that all values are different.



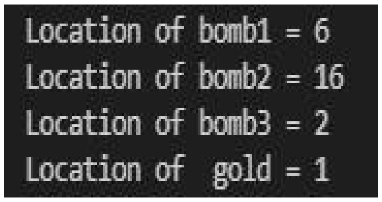
```
 /* printf("Location of bomb1 =
%d\n", bomb1);
    printf("Location of bomb2 =
%d\n", bomb2);
    printf("Location of bomb3 =
%d\n", bomb3);
    printf("Location of  gold =
%d\n", gold); */
    return 0;
}
```



```
Location of bomb1 = 5
Location of bomb2 = 8
Location of bomb3 = 4
Location of  gold = 13
```



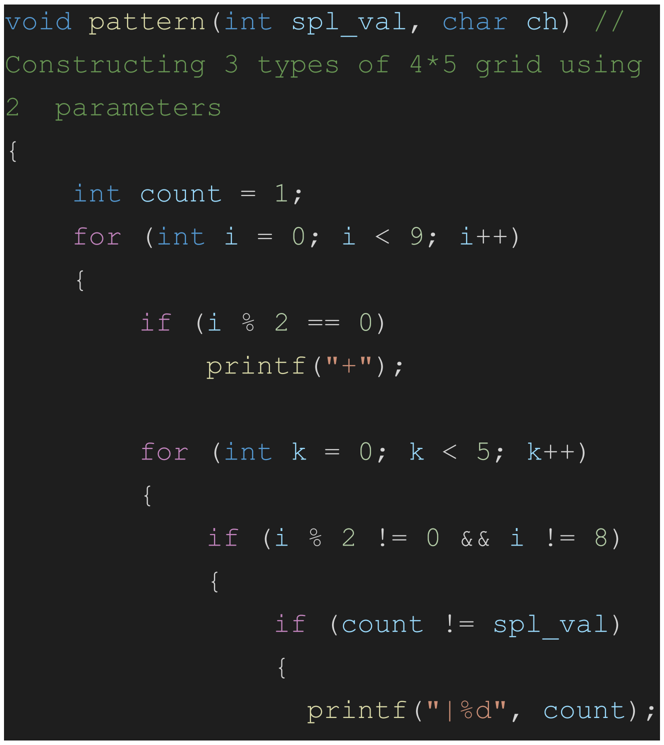
```
Location of bomb1 = 9
Location of bomb2 = 5
Location of bomb3 = 10
Location of  gold = 4
```



```
Location of bomb1 = 6
Location of bomb2 = 16
Location of bomb3 = 2
Location of  gold = 1
```

pattern(int spl_val, char ch) :

This function shows highlighted position “spl_val” (where “ch” is used to check whether the game is starting/person landed on the mine or treasure) in a 4*5 grid.



```
void pattern(int spl_val, char ch) //
Constructing 3 types of 4*5 grid using
2  parameters
{
    int count = 1;
    for (int i = 0; i < 9; i++)
    {
        if (i % 2 == 0)
            printf("+");

        for (int k = 0; k < 5; k++)
        {
            if (i % 2 != 0 && i != 8)
            {
                if (count != spl_val)
                {
                  printf("|%d", count);
```

- **Using a nested for loop, a pictorial representation of a 4*5 grid is constructed.**
- **3 types of grids are constructed (A numbered 4*5 grid,landing on a mine, landing on the box of gold).**
- **The pattern function has two**

```c
                    if (count < 10)
                        printf(" ");
                }
                else
                {
                    if (ch == 'g') //
Representing the box of gold coins grid
                        printf("|$$");
                    if (ch == 'm') //
Representing the bomb grid
                        printf("|XX");
                }
                count++;
            }
            if (i % 2 == 0)
            {
                printf("--+");
            }
        }
        if (i % 2 != 0)
        {
            printf("|");
        }
        printf("\n");
    }
    printf("\n");
}
```

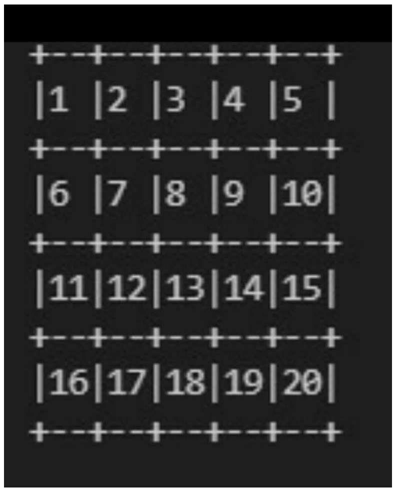
parameters (spl_val & ch), which are used to define the 3 types of grids

- ➢ **spl_val is used to provide the pattern function with the current location of the user on the grid.**

- ➢ **ch is used to trigger the respective type of grid which needs to be shown.**

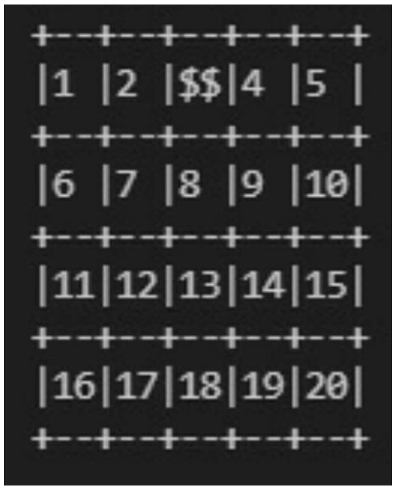
- ➢ **When landing on the mine/gold the grid constructed shows “XX”/“$$” on the current location respectively.**

pattern(0,’o’)



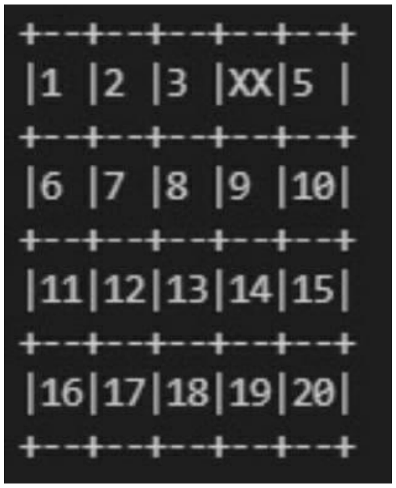
```
+--+--+--+--+--+
|1 |2 |3 |4 |5 |
+--+--+--+--+--+
|6 |7 |8 |9 |10|
+--+--+--+--+--+
|11|12|13|14|15|
+--+--+--+--+--+
|16|17|18|19|20|
+--+--+--+--+--+
```

pattern(3,’g’)



```
+--+--+--+--+--+
|1 |2 |$$|4 |5 |
+--+--+--+--+--+
|6 |7 |8 |9 |10|
+--+--+--+--+--+
|11|12|13|14|15|
+--+--+--+--+--+
|16|17|18|19|20|
+--+--+--+--+--+
```

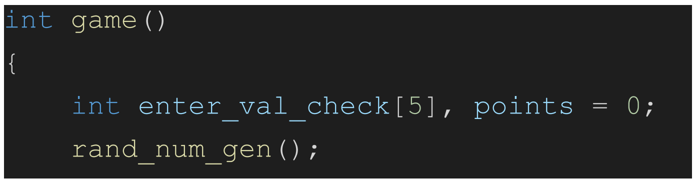
pattern(4,’m’)



```
+--+--+--+--+--+
|1 |2 |3 |XX|5 |
+--+--+--+--+--+
|6 |7 |8 |9 |10|
+--+--+--+--+--+
|11|12|13|14|15|
+--+--+--+--+--+
|16|17|18|19|20|
+--+--+--+--+--+
```

game() :

This function takes the input from the user, and then executes the code which checks whether the player has landed on a mine or on the box of gold coins, or neither of them, and keeps track of the score obtained by the player.



```
int game()
{
    int enter_val_check[5], points = 0;
    rand_num_gen();
```

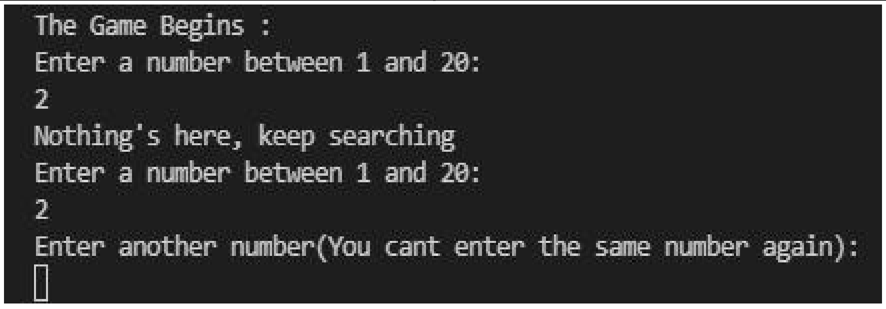
- ➢ **rand_num_gen() function is called.**

```c
    for (int i = 0; i < 5; i++)
    {
        if (i == 0)
            printf("The Game Begins
:\n"); //In the beginning of the game
        do
        {
            printf("Enter a number
between 1 and 20:\n");
            scanf("%d",
&enter_val_check[i]);
            for (int j = 0; j < i; j++)
            {
                if (enter_val_check[i]
== enter_val_check[j]) //To check if the
value was entered before
                {
                    printf("Enter
another number(You cant enter the same
number again):\n");
                    scanf("%d",
&enter_val_check[i]);
                    j = -1; //To check
from the beginning of values entered
again
                }
            }
        } while (!(enter_val_check[i] >=
1 && enter_val_check[i] <= 20)); //To
check if the values are between 1 and
20.

        if (enter_val_check[i] == bomb1
|| enter_val_check[i] == bomb2 ||
enter_val_check[i] == bomb3)
        {
            printf("Oops,You landed on a
mine!\n");
            smiley('m');
            pattern(enter_val_check[i],
'm');
```

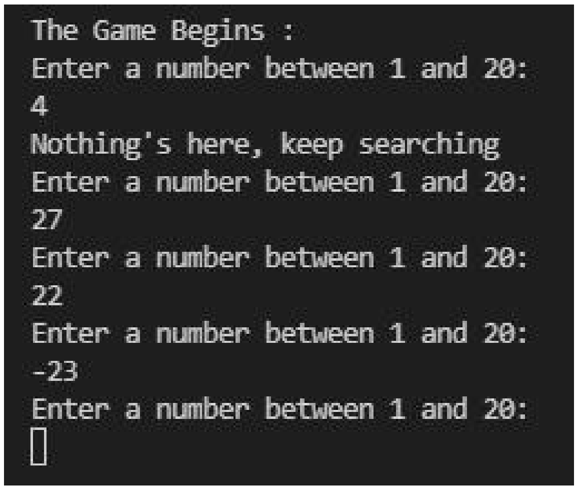
- ➢ **For loop is used to collect input from the user.**
- ➢ **for loop is used to ensure the same no. isn't entered again.**
- ➢ **do while loop to check if the number ranges from 1 to 20.**
- ➢ **if else if statement is used to check whether the user has landed on the position where the bombs/box of gold are located.**
- ➢ **If the user lands on any of the bombs, then the game is terminated with +0 points, else if he/she lands on the box of gold, he scores +100 points and the game terminates.**
- ➢ **If neither is the case then the user is provided with an extra chance with +5 points. He/she can have at most 4 extra chances.**
- ➢ **For the respective above mentioned scenarios, using respective parameters (spl_val), the pattern funct is called. To represent the 4*5 grid.**

```
            return points; //Points are being returned
        }
        else if (enter_val_check[i] == gold)
        {
            printf("$$ You are the treasure hunter $$\n");
            smiley('g');
            pattern(enter_val_check[i], 'g');
            points += 100;
            return points; //Points are being returned
        }
        else
        {
            printf("Nothing's here, keep searching\n");
            points += 5;
        }
    }
    smiley('o');
    return points; //Points are being returned
}
```

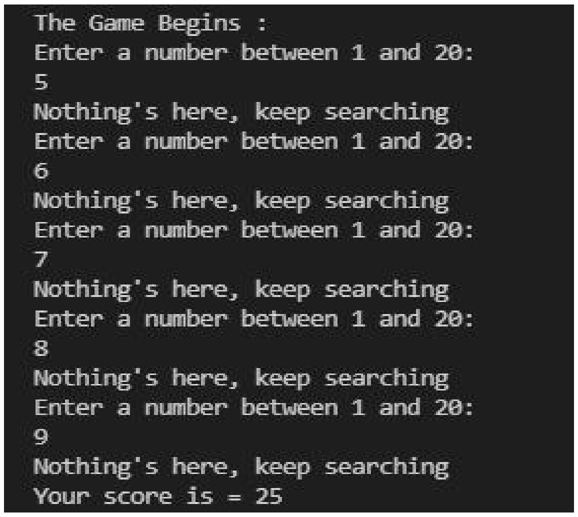
➢ **Finally the score (points) is returned**



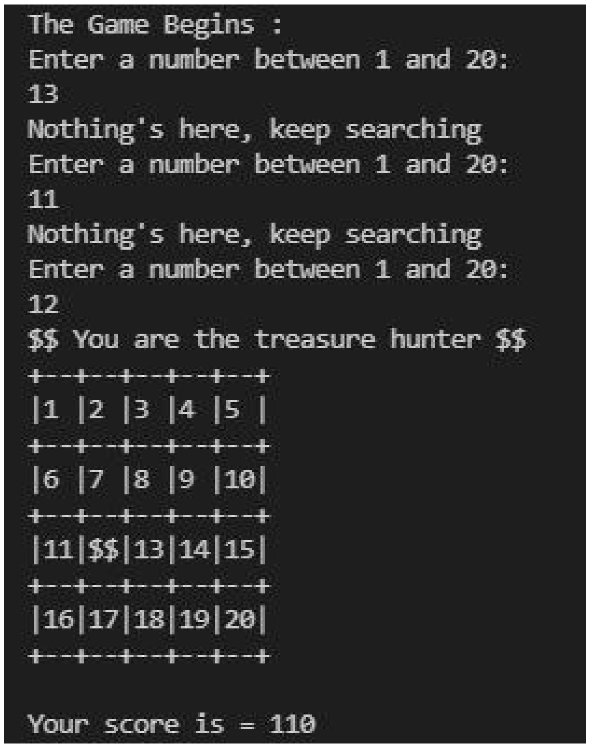
```
The Game Begins :
Enter a number between 1 and 20:
2
Nothing's here, keep searching
Enter a number between 1 and 20:
2
Enter another number(You cant enter the same number again):
```



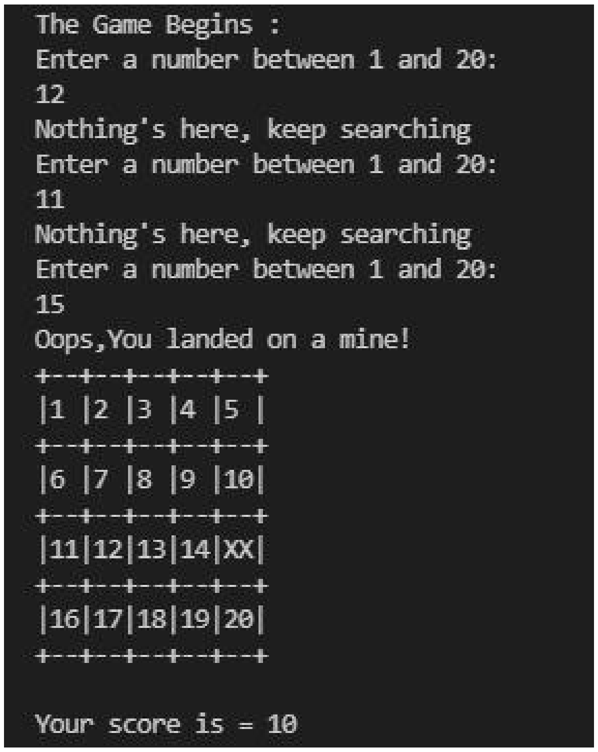
```
The Game Begins :
Enter a number between 1 and 20:
4
Nothing's here, keep searching
Enter a number between 1 and 20:
27
Enter a number between 1 and 20:
22
Enter a number between 1 and 20:
-23
Enter a number between 1 and 20:
```



```
The Game Begins :
Enter a number between 1 and 20:
5
Nothing's here, keep searching
Enter a number between 1 and 20:
6
Nothing's here, keep searching
Enter a number between 1 and 20:
7
Nothing's here, keep searching
Enter a number between 1 and 20:
8
Nothing's here, keep searching
Enter a number between 1 and 20:
9
Nothing's here, keep searching
Your score is = 25
```



```
The Game Begins :
Enter a number between 1 and 20:
13
Nothing's here, keep searching
Enter a number between 1 and 20:
11
Nothing's here, keep searching
Enter a number between 1 and 20:
12
$$ You are the treasure hunter $$
+--+--+--+--+--+
|1 |2 |3 |4 |5 |
+--+--+--+--+--+
|6 |7 |8 |9 |10|
+--+--+--+--+--+
|11|$$|13|14|15|
+--+--+--+--+--+
|16|17|18|19|20|
+--+--+--+--+--+

Your score is = 110
```



```
The Game Begins :
Enter a number between 1 and 20:
12
Nothing's here, keep searching
Enter a number between 1 and 20:
11
Nothing's here, keep searching
Enter a number between 1 and 20:
15
Oops,You landed on a mine!
+--+--+--+--+--+
|1 |2 |3 |4 |5 |
+--+--+--+--+--+
|6 |7 |8 |9 |10|
+--+--+--+--+--+
|11|12|13|14|XX|
+--+--+--+--+--+
|16|17|18|19|20|
+--+--+--+--+--+

Your score is = 10
```

main():

Program execution starts from here. This is the place where many different functions(performing various tasks) are called to complete the preparation of our game.

```
int main()
{
    print();
    char username[50];
    fgets(username, 49, stdin);
    char choice, score_change;
    int user_num_diamonds;
    do
    {
        pattern(0, 'o'); //0 and o are
used randomly, we can use any number
(except the numbers from 1 to 20) and
any alphabet (except g and m)
        int curr_score = game();
        printf("Your score is = %d\n",
curr_score);
        printf("\n");
        FILE *fp;             // need
to take care of error analysis while
pointers return null
        struct user user[50]; //need to
take care of error analysis when people
are more than 50
        int rec_size = sizeof(struct
user);
        fp =
fopen("_The_treasure_hunt_db", "ab");
//loading phase code
        if (fp == NULL)
        {
            printf("couldn't open the
file");
            return -1; //this return
only valid if loading phase is
operating after the game
        }
        fclose(fp); //k users already
present
        fp =
fopen("_The_treasure_hunt_db", "rb");
        if (fp == NULL)
```

➢ we are printing some instructions about game features and rules
We input name from the user and start the game.

➢ we call the pattern function with proper arguments for printing the grid on the terminal

➢ later we call the game function which takes cares of the tasks that need to be done while the user is playing the game and it returns points at last which we are storing as score in variable named “curr_score”

➢ we then print the curr_score of the user who played the game.

➢ we declared a file pointer (fp) and an array user in which each element is of type struct user.

➢we initialized the array user[50] , so maximum 50 different people can play.

➢ we took the size of the struct user into a variable named rec_size .

➢we appended the file called _The_treasure_hunt_db which creates the file if the file does not exist in your directory.

```
        {
            printf("couldn't open the
file");
            return -1; //this return
only valid if loading phase is
operating after the game
        }
        int num_rec_presnt =
fread(&user, rec_size, 50, fp);
        fclose(fp);
        for (int j = 0; j < 50; j++)
        {

strcpy(user[j].score_det[2].title,
"Score in the present game is");

strcpy(user[j].score_det[1].title,
"Score in the previous game is");

strcpy(user[j].score_det[0].title,
"Score in the last but two game is");
        }
        bool user_present = false;
//variable for initial state
        int i = 0;
        for (int j = 0; j <
num_rec_presnt; j++)
        {
            if (strcmp(username,
user[j].name) == 0)
            {
                user_present = true;
                i = j;
            }
        }
        /*  if user_present is true, we
need to update user[i] part of the
structure.
     if user_present is false, we need
to update user[k] part of the
structure.
```

➢we used fopen function to open a file after taking care of error analysis of the file pointer being NULL.

➢ we are then reading the details in the file into our structure and storing them .

➢ we used strcpy function to copy the username into its corresponding location in the structure.

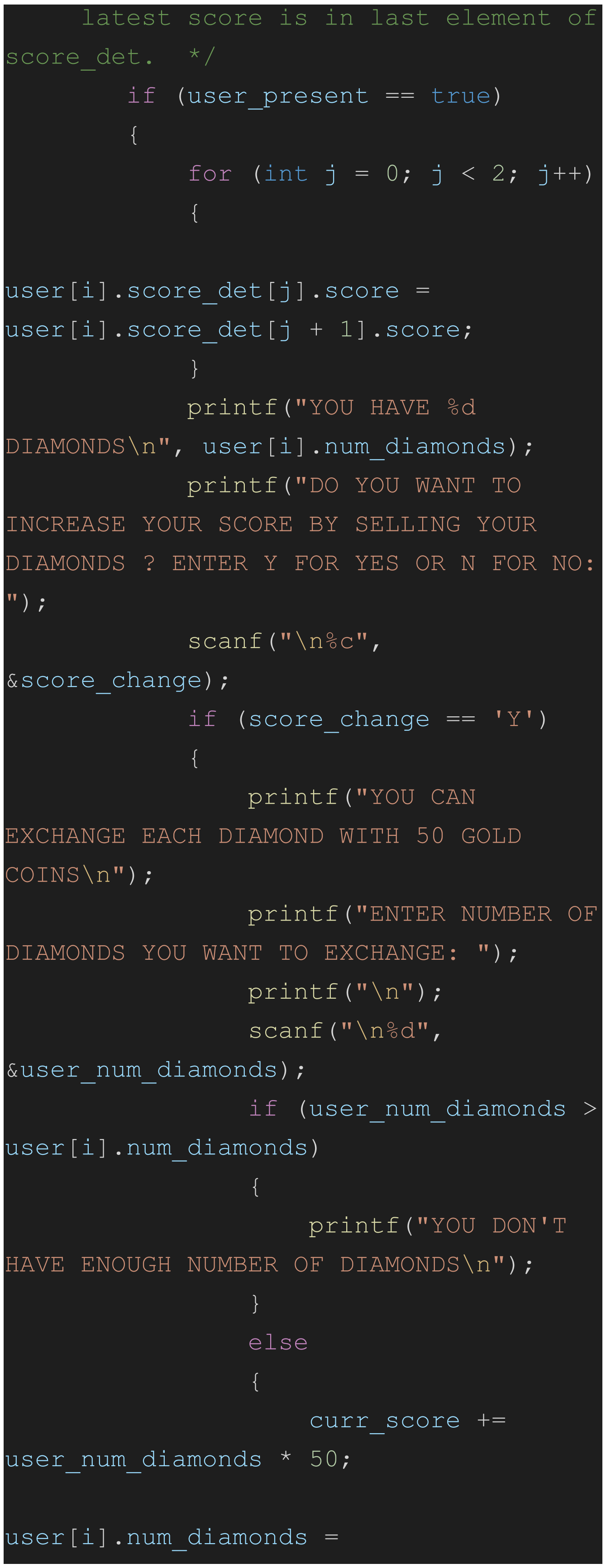
➢ we are using fread() func to read the data in the file into the structure.

➢We used the concept of state variables where two states are represented by boolean user_present which has two states true and false.

➢We are checking whether the user who is trying to play the game now already played a game or he is playing it for the first time.

➢We are checking the above task by using for loops.

➢If the user had already played a game before then we are updating the part of the structure which contains the



```
     latest score is in last element of
score_det.  */
        if (user_present == true)
        {
            for (int j = 0; j < 2; j++)
            {

user[i].score_det[j].score =
user[i].score_det[j + 1].score;
            }
            printf("YOU HAVE %d
DIAMONDS\n", user[i].num_diamonds);
            printf("DO YOU WANT TO
INCREASE YOUR SCORE BY SELLING YOUR
DIAMONDS ? ENTER Y FOR YES OR N FOR NO:
");
            scanf("\n%c",
&score_change);
            if (score_change == 'Y')
            {
                printf("YOU CAN
EXCHANGE EACH DIAMOND WITH 50 GOLD
COINS\n");
                printf("ENTER NUMBER OF
DIAMONDS YOU WANT TO EXCHANGE: ");
                printf("\n");
                scanf("\n%d",
&user_num_diamonds);
                if (user_num_diamonds >
user[i].num_diamonds)
                {
                    printf("YOU DON'T
HAVE ENOUGH NUMBER OF DIAMONDS\n");
                }
                else
                {
                    curr_score +=
user_num_diamonds * 50;

user[i].num_diamonds =
```

information about the user .

➢ We are updating the corresponding scores of the particular user

➢We used for loops for updating the scores of the particular user .

➢we are printing number of diamonds user has .

➢we are asking user if he wants to change his score by exchanging diamonds or not .

➢if the user wishes to exchange we are updating score according to the changes made else we are directly printing scores.

➢we are changing number of diamonds also after the exchanging process is done.

➢we are trying to print scores of last three games of the user so all our conditions are set for updating last three score.

➢ we are checking if the last three scores are greater than 100 .

➢if they are greater than 100 we are adding 3 diamonds to already present number of diamonds.

```
user[i].num_diamonds -
user_num_diamonds;
                }
            }
            user[i].score_det[2].score
= curr_score;
            if
(user[i].score_det[0].score >= 100 &&
user[i].score_det[1].score >= 100 &&
user[i].score_det[2].score >= 100)
            {
                user[i].num_diamonds +=
3;
            }
            for (int j = 0; j < 3; j++)
            {
                printf("%s: %d\n",
user[i].score_det[2 - j].title,
user[i].score_det[2 - j].score);
            }
        }
        else
        {

strcpy(user[num_rec_presnt].name,
username);
            for (int j = 0; j < 2; j++)
            {

user[num_rec_presnt].score_det[j].score
= -1;
            }
            printf("YOU HAVE %d
DIAMONDS\n",
user[num_rec_presnt].num_diamonds);
            printf("DO YOU WANT TO
INCREASE YOUR SCORE BY SELLING YOUR
DIAMONDS ? ENTER Y FOR YES OR N FOR NO:
");
            printf("\n");
```

➢else we are not changing the number of diamonds and continuing in the game.

➢if_else is used to check the above condition.

➢if the user is playing the game for the first time then we will be adding the details of the user to the structure without disturbing the existing data in the structure.

➢we are using for loops for appending our structure user with the new user details.

➢if the user is playing for the first time we are initializing the scores of his previous game and last but two game to -1.

➢ so showing score as -1 means the user did not play that game.

➢we are again using strcpy function to copy the username into the desired location in the structure user.

➢we are asking user for number of diamonds he wants to exchange , and after that we are checking if the user actually have that many diamonds or

```
            scanf("\n%c",
&score_change);
            if (score_change == 'Y')
            {
                printf("YOU CAN
EXCHANGE EACH DIAMOND WITH 50 GOLD
COINS\n");
                printf("ENTER NUMBER OF
DIAMONDS YOU WANT TO EXCHANGE: ");
                scanf("\n%d",
&user_num_diamonds);
                if (user_num_diamonds >
user[num_rec_presnt].num_diamonds)
                {
                    printf("YOU DON'T
HAVE ENOUGH NUMBER OF DIAMONDS\n");
                }
                else
                {
                    curr_score +=
user_num_diamonds * 50;

user[num_rec_presnt].num_diamonds =
user[num_rec_presnt].num_diamonds -
user_num_diamonds;
                }
            }

user[num_rec_presnt].score_det[2].score
= curr_score;
            for (int j = 0; j < 3; j++)
            {
                printf("%s: %d\n",
user[num_rec_presnt].score_det[2 -
j].title,
user[num_rec_presnt].score_det[2 -
j].score);
            }
        } //need to save the user
structure to the file
```

not.

➢if yes we are exchanging those diamonds each for 50 coins

➢else we are printing appropriate message.

➢ we are copying the latest score of the user from the variable curr_score which contains the score of the last game.

➢we are updating latest scores by taking num diamonds into account.

➢ we are opening the binary file in write mode.

➢ we are using fwrite() func to write the structure into the file.

➢we are writing the struct user into the file.

➢we are doing that in two different parts ,one in which is the user already played the game, other in which the user is playing the game for the first time.

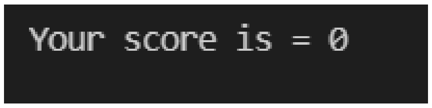
➢we are doing that by using if-else conditional statement.

```
        fp =
fopen("_The_treasure_hunt_db", "wb");

        if (user_present == true)
        {
            fwrite(&user, rec_size,
num_rec_presnt, fp);
        }
        else
        {
            fwrite(&user, rec_size,
num_rec_presnt + 1, fp);
        }
        fclose(fp);
        printf("\n");
        printf("ENTER P TO PLAY AGAIN
OR Q TO QUIT THE GAME- ");
        scanf("\n%c", &choice);
        printf("\n");
    } while (choice == 'P');
    rate();
    return 0;
}
```

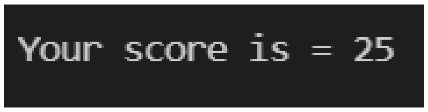
➢**after writing into the file we are closing the file.**

➢ **We are asking the user whether they want to continue playing another round or exit the game , if he wants to play again game will repeat .**

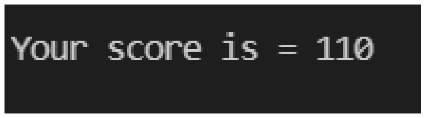
➢**else rate function is called and after execution of rate function the game ends.**



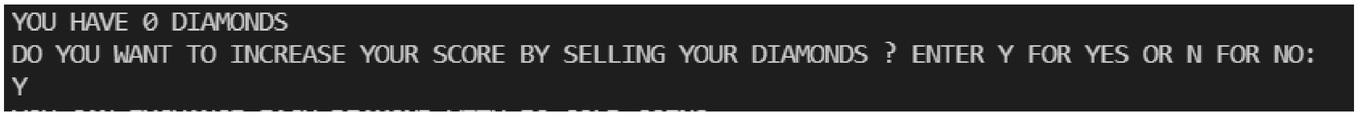
```
Your score is = 0
```



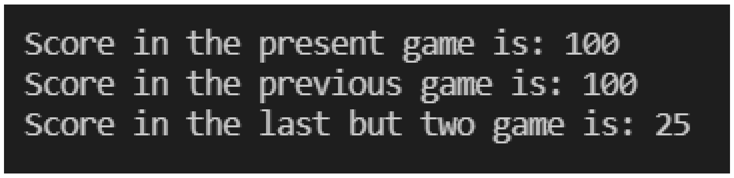
```
Your score is = 25
```



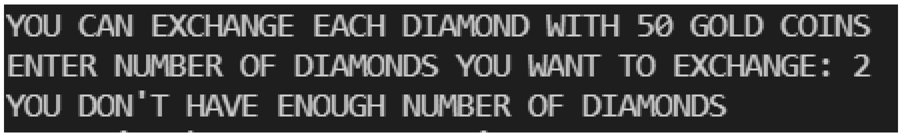
```
Your score is = 110
```



```
YOU HAVE 0 DIAMONDS
DO YOU WANT TO INCREASE YOUR SCORE BY SELLING YOUR DIAMONDS ? ENTER Y FOR YES OR N FOR NO:
Y
```



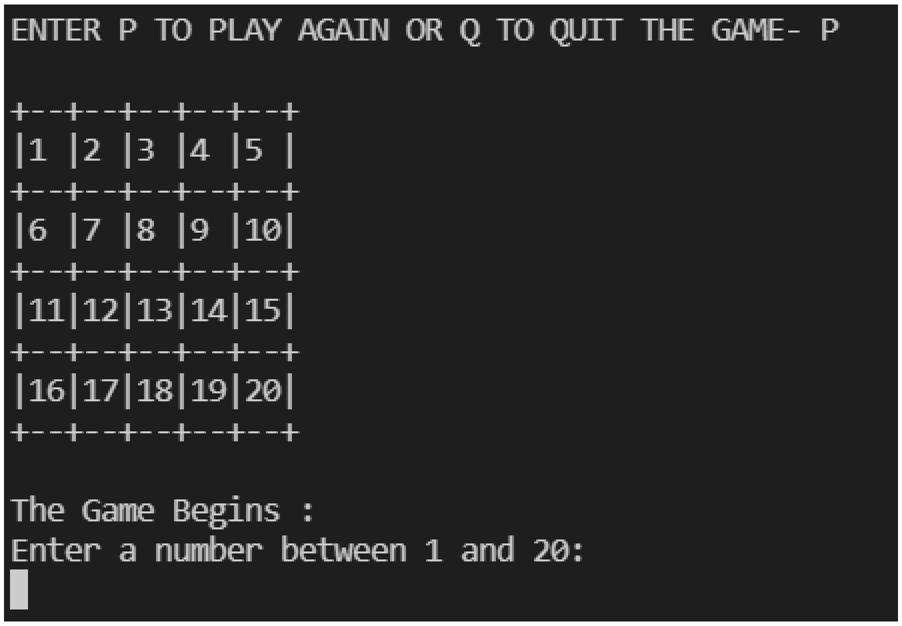
```
Score in the present game is: 100
Score in the previous game is: 100
Score in the last but two game is: 25
```



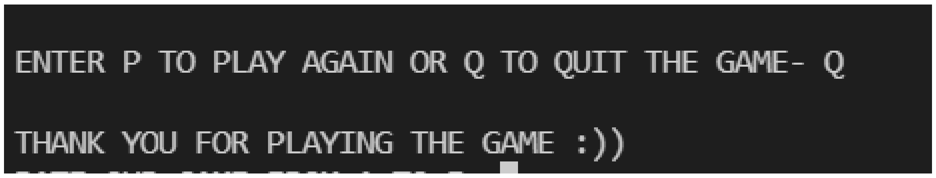
```
YOU CAN EXCHANGE EACH DIAMOND WITH 50 GOLD COINS
ENTER NUMBER OF DIAMONDS YOU WANT TO EXCHANGE: 2
YOU DON'T HAVE ENOUGH NUMBER OF DIAMONDS
```

Here we have given the user the option to continue the game or exit by using a do-while loop.

After the player enters P , it will start again with pattern() function.After the player enters Q , it will go with rate() function.

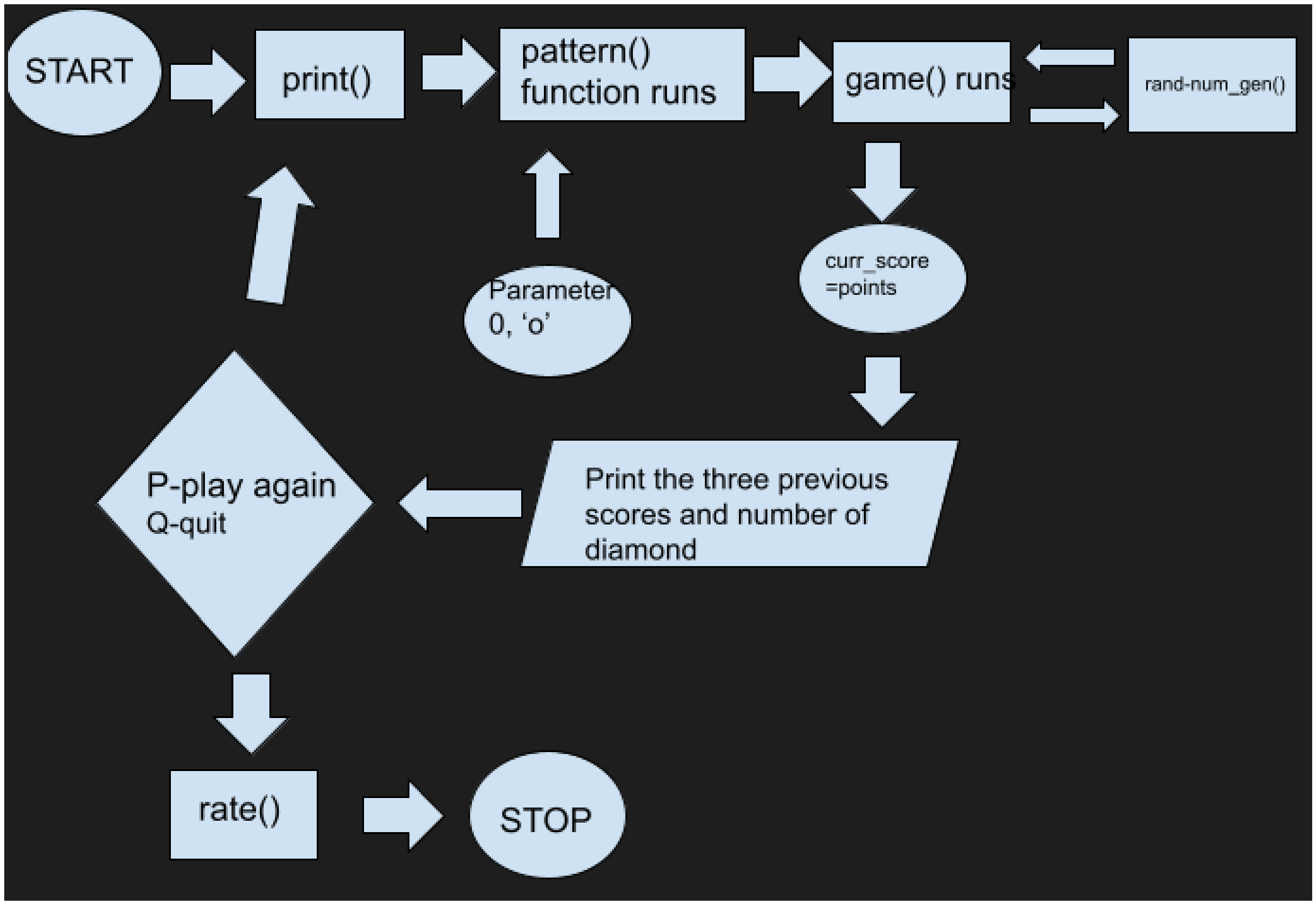


```
ENTER P TO PLAY AGAIN OR Q TO QUIT THE GAME- P

+--+--+--+--+--+
|1 |2 |3 |4 |5 |
+--+--+--+--+--+
|6 |7 |8 |9 |10|
+--+--+--+--+--+
|11|12|13|14|15|
+--+--+--+--+--+
|16|17|18|19|20|
+--+--+--+--+--+

The Game Begins :
Enter a number between 1 and 20:
```



```
ENTER P TO PLAY AGAIN OR Q TO QUIT THE GAME- Q

THANK YOU FOR PLAYING THE GAME :))
```

FLOWCHART:



In the above figure, we have demonstrated the logical sequence flow algorithm form of our game code in a flowchart format.

References:

- https://www.geeksforgeeks.org/rand-and-srand-in-ccpp for understanding working and application of rand() function
- https://stackoverflow.com/questions/48539945/behaviour-of-global-variable-in-c to understand how the value of global variables gets altered.
- https://learning.edx.org/course/course-v1:IITBombayX+CS101.1x+1T2020/home (videos) for a detailed, vivid explanation on the file handling part.
- https://www.programiz.com/c-programming/c-file-input-output helped us to develop a better understanding of types of files and other functions we can use.
- https://www.tutorialspoint.com/c_standard_library/c_function_fread.htm to understand the working of fread() function.
- CS101 lecture slides and assignment problems.