# CSE102 ASSIGNMENT 5 REPORT

## PICTURES OF THE CODE

C MuhammetFatih_Albayın.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

/* Initialize the grid by filling it with '-' */
void initializeGrid(char grid[100]) {
    int i;
    for (i = 0; i < 100; i++) {
        grid[i] = '-';
    }
}

/* Check if a ship can be placed at the given position */
int canPlaceShip(char grid[100], int length, int x, int y, int direction) {
    int i;

    for (i = 0; i < length; i++) {
        int newX = x;
        int newY = y;

        if (direction == 0) {
            newX += i; /* Horizontal placement */
        } else {
            newY += i; /* Vertical placement */
        }

        int index = newY * 10 + newX;

        /* Check if the position is valid and empty */
        if (newX >= 10 || newY >= 10 || grid[index] != '-') {
            return 0; /* Placement not possible */
        }
    }
    return 1;
}

/* Place a ship on the grid and store its coordinates */
void placeShip(char grid[100], int length, int x, int y, int direction, int shipX[4][4], int shipY[4][4], int i, FILE *fptr) {
    int j;

    for (j = 0; j < length; j++) {
        int newX = x;
        int newY = y;

        if (direction == 0) {
            newX += j;
        } else {
            newY += j;
        }

        shipX[i][j] = newX; /* Store x-coordinates */
        shipY[i][j] = newY; /* Store y-coordinates */
        fprintf(fptr, "%d %d %d\n", newX, newY, length);
    }
}

/* Generate ships and store their information */
void generateShips(char grid[100], int shipX[4][4], int shipY[4][4], int shipLengths[4]) {
    int shipLengthsTmp[4] = {4, 3, 3, 2};
    int i, x, y, direction;

    srand(time(NULL));
    FILE *fptr = fopen("ships.txt", "w");

    for (i = 0; i < 4; i++) {
        shipLengths[i] = shipLengthsTmp[i];

        int placed = 0;
        while (!placed) {
            direction = rand() % 2;
            x = rand() % 10;
            y = rand() % 10;

            /* Attempt to place the ship */
            if (canPlaceShip(grid, shipLengths[i], x, y, direction)) {
                placeShip(grid, shipLengths[i], x, y, direction, shipX, shipY, i, fptr);
                placed = 1;
            }
        }
    }
```

```c
    }

    fclose(fptr);
}

/* Print the grid to the console */
void printGrid(char grid[100]) {
    int i, j;
    printf("   0 1 2 3 4 5 6 7 8 9\n");

    for (i = 0; i < 10; i++) {
        printf("%2d ", i);
        for (j = 0; j < 10; j++) {
            printf("%c ", grid[i * 10 + j]);
        }
        printf("\n");
    }
}

/* Update the grid with the result of the move */
void updateGrid(int *x, int *y, int isHit, char grid[100]) {
    if (isHit)
        grid[(*y) * 10 + (*x)] = 'X'; /* Mark a hit */
    else
        grid[(*y) * 10 + (*x)] = 'O'; /* Mark a miss */
}

/* Check if a move hits a ship and track hits */
int checkMove(int *x, int *y, char grid[100], int shipX[4][4], int shipY[4][4], int shipLengths[4], int shipHits[4]) {
    int i, j, isHit = 0;

    for (i = 0; i < 4; i++) {
        for (j = 0; j < shipLengths[i]; j++) {
            /* Check if the coordinates match any ship's position */
            if (shipX[i][j] == *x && shipY[i][j] == *y) {
                isHit = 1; /* Hit detected */
                shipHits[i]++; /* Increment the hit count */

                /* Check if the ship is sunk */
                if (shipHits[i] == shipLengths[i]) {
                    printf("Congratulations! You sunk a %d-size ship!\n", shipLengths[i]);
                }

                return isHit;
            }
        }
    }

    return isHit;
}

/* Check if all ships are sunk */
int allShipsSunk(int shipHits[4], int shipLengths[4]) {
    int i;

    for (i = 0; i < 4; i++) {
        if (shipHits[i] < shipLengths[i]) {
            return 0; /* At least one ship is still afloat */
        }
    }

    return 1; /* All ships are sunk */
}

/* Get user input for coordinates and handle the move */
int getCoordinates(int *x, int *y, char grid[100], int shipX[4][4], int shipY[4][4], int shipLengths[4], int shipHits[4], int *moveCounter) {
    char input[20];
    int isHit;

    while (1) {
        printf("Enter coordinates (X to quit): ");
        fgets(input, sizeof(input), stdin);

        /* Check for termination input */
        if (input[0] == 'X' || input[0] == 'x') {
            printf("Terminating the program...\n");
            return 0;
        }

        /* Validate and process input */
        if (sscanf(input, "%d %d", x, y) == 2) {
            if ((*x >= 0 && *x < 10) && (*y >= 0 && *y < 10)) {
                (*moveCounter)++; /* Increment move counter */
                isHit = checkMove(x, y, grid, shipX, shipY, shipLengths, shipHits);

                if (isHit)
                    printf("HIT!\n");
                else
                    printf("MISS!\n");

                updateGrid(x, y, isHit, grid);

                /* Print the grid after each move */
                printGrid(grid);

                /* Check if all ships are sunk */
                if (allShipsSunk(shipHits, shipLengths)) {
                    printf("Congratulations! You sunk all the ships in %d moves!\n", *moveCounter);
                    return 0; /* End the game */
                }

```

```
                return 1;
            } else {
                printf("Invalid coordinates! Please enter values between 0 and 9.\n");
            }
        } else {
            printf("Invalid input, please try again!\n");
        }
    }
}

/* Main menu of the game */
/* Main menu of the game */
void menu() {
    char grid[100];
    int x, y, i, moveCounter;
    int shipX[4][4], shipY[4][4], shipLengths[4], shipHits[4];
    char playAgain;

    do {
        moveCounter = 0; /* Reset move counter */
        initializeGrid(grid);
        generateShips(grid, shipX, shipY, shipLengths);

        /* Reset ship hits to 0 */
        for (i = 0; i < 4; i++) {
            shipHits[i] = 0;
        }

        printGrid(grid);

        /* Start gameplay loop */
        while (1) {
            if (getCoordinates(&x, &y, grid, shipX, shipY, shipLengths, shipHits, &moveCounter) == 0)
                break;
        }

        /* Ask the user if they want to play again */
        printf("Press 'N' to play again or 'X' to exit: ");
        while (1) {
            scanf(" %c", &playAgain); /* Use scanf to read a single character */
            if (playAgain == 'X' || playAgain == 'x') {
                printf("Thank you for playing! Goodbye!\n");
                return; /* Exit the menu */
            } else if (playAgain == 'N' || playAgain == 'n') {
                printf("Starting a new game...\n");
                break; /* Exit the inner loop and restart the game */
            } else {
                printf("Invalid input. Press N for a new game or X to exit: ");
            }
        }
    } while (1); /* Repeat the game until the user chooses to exit */
}

/* Entry point of the program */
int main() {
    printf("Welcome to the Battleship Game!\n");

    menu();

    return 0;
}
```

# THE GENERATED OUTPUTS

```
albay@albay-VirtualBox:~/Desktop$ gcc -ansi MuhammetFatih_Albayın.c  -o  m
albay@albay-VirtualBox:~/Desktop$ ./m
Welcome to the Battleship Game!
   0 1 2 3 4 5 6 7 8 9
 0 - - - - - - - - - -
 1 - - - - - - - - - -
 2 - - - - - - - - - -
 3 - - - - - - - - - -
 4 - - - - - - - - - -
 5 - - - - - - - - - -
 6 - - - - - - - - - -
 7 - - - - - - - - - -
 8 - - - - - - - - - -
 9 - - - - - - - - - -
Enter coordinates (X to quit): 7 6
HIT!
   0 1 2 3 4 5 6 7 8 9
 0 - - - - - - - - - -
 1 - - - - - - - - - -
 2 - - - - - - - - - -
 3 - - - - - - - - - -
 4 - - - - - - - - - -
 5 - - - - - - - - - -
 6 - - - - - - - X - -
 7 - - - - - - - - - -
 8 - - - - - - - - - -
 9 - - - - - - - - - -
Enter coordinates (X to quit): 7 7
HIT!
```

```
Enter coordinates (X to quit): 7 8
HIT!
   0 1 2 3 4 5 6 7 8 9
 0 - - - - - - - - - -
 1 - - - - - - - - - -
 2 - - - - - - - - - -
 3 - - - - - - - - - -
 4 - - - - - - - - - -
 5 - - - - - - - - - -
 6 - - - - - - - X - -
 7 - - - - - - - X - -
 8 - - - - - - - X - -
 9 - - - - - - - - - -
Enter coordinates (X to quit): 7 9
Congratulations! You sunk a 4-size ship!
HIT!
   0 1 2 3 4 5 6 7 8 9
 0 - - - - - - - - - -
 1 - - - - - - - - - -
 2 - - - - - - - - - -
 3 - - - - - - - - - -
 4 - - - - - - - - - -
 5 - - - - - - - - - -
 6 - - - - - - - X - -
 7 - - - - - - - X - -
 8 - - - - - - - X - -
 9 - - - - - - - X - -
Enter coordinates (X to quit): 2 6
HIT!
```

```
Congratulations! You sunk a 2-size ship!
HIT!
   0 1 2 3 4 5 6 7 8 9
 0 - - - - - - - - - -
 1 - - - - - - - - - -
 2 - - - - - - - - - -
 3 - - - - - - - - - -
 4 - - - - - - - - - -
 5 X X - X X X - - - -
 6 - - X - - - - X - -
 7 - - X - - - - X - -
 8 - - X - - - - X - -
 9 - - - - - - - X - -
Congratulations! You sunk all the ships in 12 moves!
Press 'N' to play again or 'X' to exit: x
Thank you for playing! Goodbye!
```

PICTURES OF THE GENERATED FILES

```
1  Shot: 5 5 - MISS
2  Shot: 3 3 - MISS
3  Shot: 2 2 - MISS
4  Shot: 1 1 - MISS
5  Shot: 9 9 - MISS
6  Shot: 8 8 - MISS
7  Shot: 7 7 - MISS
8  Shot: 6 6 - MISS
9  Shot: 5 5 - MISS
10 Shot: 4 4 - HIT
11 Shot: 0 0 - MISS
12 Shot: 4 3 - HIT
13 Shot: 4 2 - HIT
14 Shot: 4 1 - MISS
15 Shot: 4 5 - HIT
```

```
1  7 6 4
2  7 7 4
3  7 8 4
4  7 9 4
5  2 6 3
6  2 7 3
7  2 8 3
8  3 5 3
9  4 5 3
10 5 5 3
11 0 5 2
12 1 5 2
```