```
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
#include<strings.h>
#define MINBATTLEZONEHEIGHT 6
#define MINBATTLEZONEWIDTH  6
#define MAXBATTLEZONEHEIGHT 20
#define MAXBATTLEZONEWIDTH  20
#define MINFLEETSIZE       1

#define COMPUTERSHIP ' '
#define USERSHIP 'U'
#define BOMBED 'X'
#define MISS 'O'

#define PARAM_OUT_OF_BOUNDS 2

int zoneWidth, zoneHeight, fleetSize, userVictory, x, y, j, k;
char battleZone[MAXBATTLEZONEWIDTH][MAXBATTLEZONEHEIGHT];
int getIntInRange(int bot, int top){
        //Get an Integer from the input and
        //return it, if it is in the range bot to top, inclusive

        int val;

        printf("\n(%d to %d) > : ", bot, top);
        scanf("%d", &val);

        while(val < bot || val > top){
                printf("\nOut of Range\nTry Again: ");
                scanf("%d", &val);
```

```
        }
        return val;
}

int initBattleZoneWidth(){
        printf("\nEnter Valid Battle Zone Width");
        return getIntInRange(MINBATTLEZONEWIDTH, MAXBATTLEZONEWIDTH);
}

int initBattleZoneHeight(){
        printf("\nEnter Valid Battle Zone Height");
        return getIntInRange(MINBATTLEZONEHEIGHT, MAXBATTLEZONEHEIGHT);
}

int initFleetSize(int max){

        printf("\nEnter Fleet Size");
        return getIntInRange(MINFLEETSIZE, max);
}




void setUp(){
        //Initialize Battlezone Width
        //Initialize Battlezone Height
        //Initialize Fleet size
        //Initialize the BattleZone with "empty sea"
        //Initialize Random Number seed.

        battleZone[j][k] = 0;
        zoneWidth  = initBattleZoneWidth();
```

```
	zoneHeight = initBattleZoneHeight();
	fleetSize  = initFleetSize((zoneWidth*zoneHeight)/2);

}

void assert(int condition, char * message){
	if (!condition){
		printf("%s", message);
		exit(PARAM_OUT_OF_BOUNDS);
	}
}

void printBattleZone(int x, int y){
	//Print a portio of the BattleZone from (0,0) to (x,y)
	//Ensure parameters are legal using assert
	assert(x <= zoneWidth, "\nprintBattleZone: parameter > zoneWidth\n");
	assert(y <= zoneHeight,"\nprintBattleZone: parameter > zoneHeight\n");

	printf("\n\ny |");

		for(j=0; j<zoneWidth; j++){
			printf("\t%d", j+1);
		}
		printf(" - x");
		printf("\n\n");
		for(j=0; j<zoneHeight; j++){
			printf("%d |\t", j+1);
			for(k=0; k<zoneWidth; k++){
				printf("%c\t", battleZone[j][k]);
			}
		printf("\n\n");
```

```
                }
}



void placeMyShips(){
        int i, j, k, c;
        printf("\n\nEnter ship co-ordinates:");
        for(i=0; i<fleetSize; i++){
                printf("\n\nShip %d:\nx,y > ", i+1);
                scanf("%d,%d", &k,&j);
                fflush(stdin);

                        if(battleZone[j-1][k-1]==USERSHIP){
                                printf("Overwrite detected. Please Try again.\n");
                                i=i-1;
                                }else{
                                        battleZone[j-1][k-1] = USERSHIP;
                                        printBattleZone(zoneWidth, zoneHeight);
                                }

                        if(j-1 >= zoneHeight || k-1 >= zoneWidth || j-1<0 || k-1<0){
                                printf("Please enter valid values.\n");
                                i=i-1;
                                }
        }
}
```

```
	//Ask user for fleet size number of x,y coordinates
	//so that the ships can be placed in the BattleZone
	//Ensure user not overwriting previously placed ship
	//Placing a ship, means writing a character to represent the user's
	//ship into the grid at x,y



void placeComputerShips(){
	int i, j, k;
	srand(time(NULL));
	printf("Loading.. Please Wait.");
	for(i=0; i<fleetSize; i++){
		sleep(1);					//Same function as the one below, however this one is for possible repeats of the loop. (Some computers are very fast - I ran into issues without these)
		j = rand()%(zoneHeight);
		sleep(1);					//To ensure the same time value is not read - increasing randomness and ensuring
		k = rand()%(zoneWidth);
		if(battleZone[j][k] == USERSHIP || battleZone[j][k] == COMPUTERSHIP || battleZone[j][k] == MISS){
			i=i-1;
		}else{
		battleZone[j][k] = COMPUTERSHIP;
		}
	}
		printBattleZone(zoneWidth, zoneHeight);


}
	//Place fleet size number of ships.
	//It will generate a random number, x, between 0 and zoneWidth
```

```
	//for the x ordinate.
	//It will generate a random number, y, between 0 and zoneHeigh
	//for the y ordinate.
	//Check that there is no ship at x,y and place a ship there.
	//If a ship already exists at x,y then choose again.
	//Placing a ship, means writing a character to represent the computer's
	//ship into the grid at x,y


void checkBoard(){
	int v, r;
	v=0;
	r=0;
	for(j=0; j<=zoneHeight; j++){
		for(k=0; k<=zoneWidth; k++){

			if(battleZone[j][k] == USERSHIP){
				v=v+1;
			}

			if(battleZone[j][k] == COMPUTERSHIP){
				r=r+1;
			}

		}
	}
	if(v==0){
		userVictory=0;
	}
	if(r==0){
		userVictory=1;
	}
```

```
}

void play(){
	int i, j, k, w;
	printf("Battle commencing. Make your mark.");
	while(1){

	//Player's Turn

				userVictory = 2; //Prevents automatic defeat.
				w=1;
				while(w>=1){
				printf("\n\nEnter a co-ordinate to bomb.\nx,y > ");
				scanf("%d,%d", &k, &j);
				fflush(stdin);
				if(j-1 > zoneHeight || k-1 > zoneWidth || j-1<0 || k-1<0){
					printf("Please enter valid values.\n");
					w++;
				}
				if(battleZone[j-1][k-1]==USERSHIP){
					w++;
					printf("\nNo point in bombing yourself. Try again.\n");
				}
				w--;
				}


				if(battleZone[j-1][k-1]==USERSHIP){
					i=i+1;
					printf("\nNo point in bombing yourself. Try again.\n");
				}
```

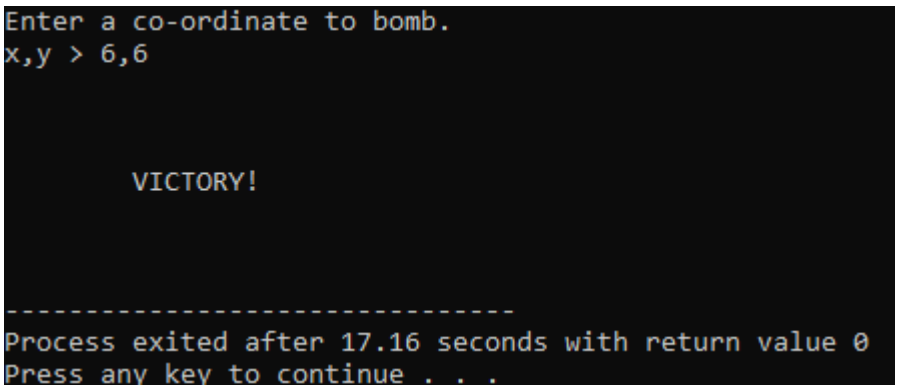
```
                if(battleZone[j-1][k-1]==COMPUTERSHIP){
                        battleZone[j-1][k-1] = BOMBED;
                }else{
                        battleZone[j-1][k-1] = MISS;
                }

        checkBoard();

        if(userVictory==1){
                printf("\n\n\n\tVICTORY!\n\n\n");
                break;
        }
        if(userVictory==0){
                printf("\n\n\n\tDEFEAT.\n\n\n");
                break;
        }

                printf("Computer's Turn..");
                sleep(1);

//Computer's Turn

        j = rand()%(zoneWidth);
        k = rand()%(zoneHeight);
                if(battleZone[j][k]==USERSHIP||battleZone[j][k]==COMPUTERSHIP){
                        battleZone[j][k] = BOMBED;
                }
                else{
                        battleZone[j][k] = MISS;
                }
```

```
                printBattleZone(zoneWidth, zoneHeight);
                printf("Computer placed.\n");

                checkBoard();

                if(userVictory==1){
                        printf("\n\n\n\tVICTORY!\n\n\n");
                        break;
                }
                if(userVictory==0){
                        printf("\n\n\n\tDEFEAT.\n\n\n");
                        break;
                }
        }
}


int main(){

        setUp();
        printBattleZone(zoneWidth, zoneHeight);
        placeMyShips();
        placeComputerShips();
        play();
        return(0);
}
```

```
Enter Valid Battle Zone Width
(6 to 20) > : 10

Enter Valid Battle Zone Height
(6 to 20) > : 10

Enter Fleet Size
(1 to 50) > : 2


y |     1       2       3       4       5       6       7       8       9       10 - x

1 |

2 |

3 |

4 |

5 |

6 |

7 |

8 |

9 |

10 |


Enter ship co-ordinates:

Ship 1:
x,y > 1,1

y |     1       2       3       4       5       6       7       8       9       10 - x

1 |     U

2 |

3 |

4 |

5 |

6 |

7 |

8 |

9 |

10 |


Ship 2:
x,y > 5,5

y |     1       2       3       4       5       6       7       8       9       10 - x

1 |     U

2 |

3 |

4 |

5 |                                     U

6 |

7 |

8 |

9 |

10 |

Loading.. Please Wait.

y |     1       2       3       4       5       6       7       8       9       10 - x

1 |     U

2 |

3 |

4 |

5 |                                     U

6 |

7 |

8 |

9 |

10 |

Battle commencing. Make your mark.
```

When a computer ship is bombed it displays X

When all computer ships are bombed the following is shown:



There is also an equivalent for Defeat.