```cpp
#include "libRecipe.h"

#include <cctype>
#include <ctime>

using namespace RecipeSpace;

int main(int argc, char** argv)
{
    srand(time(nullptr));

    if(argc != 7){
        cerr << "Wrong number of arguments: <Rows> <Cols> <Bush Chance> <Trees> <Flint Stones> <Turns>" << endl;
        exit(ERROR_COUNT);
    }

    int intRows = convStrToInt(argv[1]);
    int intCols = convStrToInt(argv[2]);
    int intBushChance = convStrToInt(argv[3]);
    int intNoTrees = convStrToInt(argv[4]);
    int intNoFlintStones = convStrToInt(argv[5]);
    int intNoTurns = convStrToInt(argv[6]);

    //Create GAME
    GameWorld theGame = createGame(intRows, intCols, intNoTrees, intNoFlintStones, intBushChance, intNoTurns);

    char chInput = '\0';
    bool blnContinue = true;

    do{
        system("cls");
        printGame(theGame);
        cin >> chInput;
        chInput = tolower(chInput);
        switch(chInput){
        case 'a':
        case 'd':
        case 'w':
        case 'x':
        case 'q':
        case 'e':
        case 'z':
        case 'c':
            {
                movePlayer(theGame, chInput);
                break;
            }
        case 'j':
            {
                perfomTask(theGame, CHOP_WOOD);
                break;
            }
        case 'k':
            {
                perfomTask(theGame, CRAFT_ITEM);
                break;
            }
        case 'l':
            {
                perfomTask(theGame, LIGHT_FIRE);
                break;
            }
        case 'p':
            {
                blnContinue = false;
                break;
            }
        default:
            {
                cerr << "Please enter the correct characters" << endl;
            }
        }

        //Check Game State
        checkGameState(theGame);

        if(theGame.state == WON || theGame.state == LOST){
            blnContinue = false;
        }

    }while(blnContinue);

    //Check how the game exited
    cout << "=================================================================" << endl;
    if(theGame.state == WON){
        cout << "You won the game, you lit the fire" << endl;
    }else if(theGame.state == LOST){
        cout << "You lost the game, run out of turns before darkness" << endl;
    }
    cout << "=================================================================" << endl;


    //Free Memory, give it back to the free store
    deallocMemory(theGame.arrGame, theGame.dimGameWorld.intRows);

    return 0;
}

```

```cpp
#ifndef LIBRECIPE_H_INCLUDED
#define LIBRECIPE_H_INCLUDED

#include <iostream>
#include <cassert>

using namespace std;

namespace RecipeSpace{

    enum gameStatus{
        SUCCESS, ERROR_CONV, ERROR_COUNT
    };

    enum gameState{
        RUNNING, LOST, WON
    };

    enum gameFeatures{
        SPACE, PLAYER, BUSH, TREE, FLINT, FIRE, PLAYER_TREE
    };

    enum playerTasks{
        CRAFT_ITEM, CHOP_WOOD, LIGHT_FIRE
    };

    const char CHARACTERS[7] = {'.', 'P', 'B', 'T', 'F', '0','P'};

    typedef int* OneDArray;
    typedef OneDArray* TwoDArray;

    struct Position{
        int intRow;
        int intCol;
    };

    struct Dimension{
        int intRows;
        int intCols;
    };

    struct GameWorld{
//        int intRows;
//        int intCols;
        Dimension dimGameWorld;

        Position posPlayer;
        TwoDArray arrGame;
        int numSticks;
        int numFlints; //Collected
        int numLogs;
        int numAxes;
        int numFireKits;
        int numTurns;
        gameState state;
    };

    //Functions
    int convStrToInt(string strNum);
    int generateRandomNum(int intLower, int intUpper);
    GameWorld createGame(int intRows, int intCols, int numTrees, int numFlint, int bushChance, int numTurns);
    void printGame(const GameWorld& myGame);
    void movePlayer(GameWorld& myGame, char chInput);
    void perfomTask(GameWorld& MyGame, int intInput);
    void checkGameState(GameWorld& myGame);
    void deallocMemory(TwoDArray& arrGame, int intRows);
}

#endif // LIBRECIPE_H_INCLUDED
```

```cpp
#include "libRecipe.h"
#include <sstream>

namespace RecipeSpace{

    int convStrToInt(string strNum){
        int intNum = 0;
        stringstream ss {strNum};
        ss >> intNum;
        if(ss.fail()){
            cerr << "Failed to convert from string to integer" << endl;
            exit(ERROR_CONV);
        }
        return intNum;
    }

    int generateRandomNum(int intLower, int intUpper){
        int intRange = intUpper - intLower + 1;
        return rand() % intRange + intLower;
    }

    void placeFeature(GameWorld myGame, int intFeature, int intFeatureCount){
        int intRow = 0;
        int intCol = 0;
        for(int i = 0; i < intFeatureCount; i++){
            intRow = generateRandomNum(0, myGame.dimGameWorld.intRows - 1);
            intCol = generateRandomNum(0, myGame.dimGameWorld.intCols - 1);
            while(myGame.arrGame[intRow][intCol] != SPACE){
                intRow = generateRandomNum(0, myGame.dimGameWorld.intRows - 1);
                intCol = generateRandomNum(0, myGame.dimGameWorld.intCols - 1);
            }
            myGame.arrGame[intRow][intCol] = intFeature;
        }
    }

    TwoDArray initGame(int intRows, int intCols){
        TwoDArray arrGame;
        arrGame = new OneDArray[intRows];
        for(int r = 0; r < intRows; r++){
            arrGame[r] = new int[intCols];
            for(int c = 0; c < intCols; c++){
                arrGame[r][c] = SPACE;
            }
        }
        return arrGame;
    }

    GameWorld createGame(int intRows, int intCols, int numTrees, int numFlint, int bushChance,
int numTurns){
        GameWorld myGame;
        myGame.dimGameWorld = {intRows, intCols};
        myGame.state = RUNNING;

        //Allocate Memory and initialize values to space
        myGame.arrGame = initGame(intRows, intCols);

        //Place Game Characters/Features

        //Player Struct
        myGame.posPlayer.intRow = intRows/2;
        myGame.posPlayer.intCol = intCols/2;
        //Player in Array
        myGame.arrGame[myGame.posPlayer.intRow][myGame.posPlayer.intCol] = PLAYER;

        // Trees
        placeFeature(myGame, TREE, numTrees);
        // Flints
        placeFeature(myGame, FLINT, numFlint);
        // Bushes
        for(int r = 0; r < intRows; r++){
            for(int c = 0; c < intCols; c++){
                int intChance = generateRandomNum(0, 100);
                if(intChance <= bushChance && myGame.arrGame[r][c] == SPACE){
                    myGame.arrGame[r][c] = BUSH;
                }
            }
        }

        //Units of items
        myGame.numSticks = 0;
        myGame.numFlints = 0;
        myGame.numLogs = 0;
        myGame.numAxes = 0;
        myGame.numFireKits = 0;

        //Turns left
        myGame.numTurns = numTurns;

        return myGame;
    }

    void printGame(const GameWorld& myGame){
        for(int r = 0;   r < myGame.dimGameWorld.intRows; r++){
            for(int c = 0; c < myGame.dimGameWorld.intCols; c++){
                int index = myGame.arrGame[r][c];
                cout << CHARACTERS[index] << " ";
            }
            cout << endl;
        }

        //Menu
        cout << endl;
        cout << "=================================================" << endl;
        cout << "A) Left D) Right W) uP X) Down Q) Top Left E) Top Right Z) Bottom Left C) Bottom Right" << endl;
        cout << "P) Quit Game " << endl;
        cout << "=================================================" << endl;
        cout << "Other actions: J) Chop Wood K) Craft an Item L) Light a Fire" << endl;
        cout << "=================================================" << endl;
        cout << "Items carried => Sticks: " << myGame.numSticks << " Flints: " << myGame.numFlints << " Logs: " << myGame.numLogs <<
                " Axes: " << myGame.numAxes << " Fire Kits: " << myGame.numFireKits << endl <<
                "Turns Left: " << myGame.numTurns << endl;
    }

    bool isValid(int intRows, int intCols, int intDR, int intDC){
        return (intDR >= 0 && intDR < intRows && intDC >= 0 && intDC < intCols);
    }

    void movePlayer(GameWorld& myGame, char chInput){
        int intDR = myGame.posPlayer.intRow;
        int intDC = myGame.posPlayer.intCol;

        switch(chInput){
        case 'a':
            {
                intDC--;
                break;
            }
        case 'd':
            {
                intDC++;
                break;
            }
        case 'w':
            {
                intDR--;
                break;
            }
        case 'x':
            {
                intDR++;
                break;
            }
        case 'q':
            {
                intDR--;
                intDC--;
                break;
            }
        case 'e':
            {
                intDR--;
                intDC++;
                break;
            }
        case 'z':
            {
                intDR++;
                intDC--;
                break;
            }
        case 'c':
            {
                intDR++;
                intDC++;
                break;
            }
        }

        //Valid move? In bounds?
        if(isValid(myGame.dimGameWorld.intRows, myGame.dimGameWorld.intCols, intDR, intDC)){
            //Set the destination feature and update item inventory
            if(myGame.arrGame[intDR][intDC] == BUSH){
                myGame.numSticks++;
                myGame.arrGame[intDR][intDC] = PLAYER;
            }else if(myGame.arrGame[intDR][intDC] == FLINT){
                myGame.numFlints++;
                myGame.arrGame[intDR][intDC] = PLAYER;
            }else if(myGame.arrGame[intDR][intDC] == TREE){
                myGame.arrGame[intDR][intDC] = PLAYER_TREE;
            }else if(myGame.arrGame[intDR][intDC] == SPACE){
                myGame.arrGame[intDR][intDC] = PLAYER;
            }

            //Update the source location
            if(myGame.arrGame[myGame.posPlayer.intRow][myGame.posPlayer.intCol] == PLAYER_TREE){
                myGame.arrGame[myGame.posPlayer.intRow][myGame.posPlayer.intCol] = TREE;
            }else{
                myGame.arrGame[myGame.posPlayer.intRow][myGame.posPlayer.intCol] = SPACE;
            }


            myGame.posPlayer = {intDR, intDC};
            myGame.numTurns--;
        }
    }

    void perfomTask(GameWorld& myGame, int intInput){
        if(intInput == CHOP_WOOD){
            if(myGame.numAxes >= 1){
                //Remove trees in a one square radius of player
                for(int r = myGame.posPlayer.intRow - 1; r <= myGame.posPlayer.intRow + 1; r++){
                    for(int c = myGame.posPlayer.intCol - 1; c <= myGame.posPlayer.intCol + 1; c++){
                        if(isValid(myGame.dimGameWorld.intRows, myGame.dimGameWorld.intCols, r, c) && myGame.arrGame[r][c] == TREE){
                            //Add one log to players collection for each tree
                            myGame.numLogs++;
                            //Remove tree
                            myGame.arrGame[r][c] = SPACE;
                        }
                    }
                }
            }
        }else if(intInput == CRAFT_ITEM){
            //Must carry right amount of materials
            if(myGame.numSticks >= 1 && myGame.numFlints >= 1){
                //Create axe
                myGame.numAxes++;
                //Decrement materials
                myGame.numSticks--;
                myGame.numFlints--;
            }
            if(myGame.numSticks >= 2 && myGame.numFlints >= 1 && myGame.numLogs >= 3){
                //Create fire kit
                myGame.numFireKits++;
                //Decrement materials
                myGame.numSticks -= 2;
                myGame.numFlints -= 1;
                myGame.numLogs -= 3;
            }
        }else if(intInput == LIGHT_FIRE){
            if(myGame.numFireKits >= 1){
                //Light fire in gameworld position nearby
                // Assumption: Fire can burn anything
                int intRow = generateRandomNum(-1, 1) + myGame.posPlayer.intRow;
                int intCol = generateRandomNum(-1, 1) + myGame.posPlayer.intCol;
                while(!isValid(myGame.dimGameWorld.intRows, myGame.dimGameWorld.intCols, intRow, intCol)){
                    intRow = generateRandomNum(-1, 1) + myGame.posPlayer.intRow;
                    intCol = generateRandomNum(-1, 1) + myGame.posPlayer.intCol;
                }
                myGame.arrGame[intRow][intCol] = FIRE;

                //Maybe, reduce on the firekit numbers
                myGame.numFireKits--;
                //Won game
                myGame.state = WON;
            }
        }
    }

    void checkGameState(GameWorld& myGame){
        if(myGame.numTurns == 0){
            myGame.state = LOST;
        }
    }

    void deallocMemory(TwoDArray& arrGame, int intRows){
        assert(arrGame != nullptr);
        for(int r = 0; r < intRows; r++){
            delete [] arrGame[r];
        }
        delete [] arrGame;
        arrGame = nullptr;
    }
}

```