```
#include <iostream>
#include <vector>
#include <string>
#include <fstream>
#include <cassert>

using namespace std;


vector<string> getWordsFromLine(string line)
{
	vector<string> parts;

	int prevPos = 0;
	int spacePos = line.find(" ", 0);

	string word;

	while (spacePos != std::string::npos)
	{
		word = line.substr(prevPos, (spacePos-prevPos));

		if (!word.empty()) parts.push_back(word);

		prevPos = spacePos + 1;
		spacePos = line.find(" ", prevPos);
	}

	word = line.substr(prevPos, (spacePos-prevPos));

	if (!word.empty()) parts.push_back(word);

	return parts;
}

void ProcessInventoryAction(vector<string> & parts, vector<string> & playerInv, vector<string> & roomInv)
{
	if (parts.size() < 2) return;

	string item = "";
	for (int i=1; i < parts.size(); i++)
	{
		item += parts[i];
		if (i < parts.size() - 1)
		{
			item += " ";
		}
	}

	vector<string>::iterator it;

	if (parts[0] == "create")
	{
		cout << "You magically create a " << item << "\n";
		roomInv.push_back(item);
	}

	if (parts[0] == "drop" || parts[0] == "d")
	{
		// cout << "dropping " << parts[1] << endl;
		it = find(playerInv.begin(), playerInv.end(), item);
		if (it == playerInv.end())
		{
			cout << "You don't have a " << item << "\n";
		}
		else
		{
			cout << "You drop the " << item << "\n";
			// remove from inventory
			playerInv.erase(it);

			// add to room
			roomInv.push_back(item);
		}
	}
	if (parts[0] == "take" || parts[0] == "t")
	{
		// cout << "taking " << parts[1] << endl;
		it = find(roomInv.begin(), roomInv.end(), item);
		if (it == roomInv.end())
		{
			cout << "There's no " << item << " here\n";

		}
		else
		{
			cout << "You pick up the " << item << "\n";
			roomInv.erase(it);
			playerInv.push_back(item);

		}
	}
}

void printVec(vector<string> & v)
{
	for (int i=0; i<v.size(); i++)
	{
		cout << v[i];
		if (i < v.size()-1) cout << ",";
	}
}

void InitRooms(vector<string> & names, vector<vector<string> > & contents, vector<string * > & exits, vector<string> &
directions)
{
	ifstream roomFile("room_map.txt");
	assert(roomFile);

	string line;

	string roomName;

	string *roomExits;
	vector<string> roomContents;

	while (getline(roomFile, line))
	{
		if (line.empty()) continue;
		line.erase(remove(line.begin(), line.end(), '\t'), line.end());

		vector<string> parts = getWordsFromLine(line);

		if (parts[0] == "room")
		{
			roomExits = new string[6];
			roomContents.clear();
		}
		else if (parts[0] == "}")
		{
			// finalize room
			if (roomName.empty())
			{
				cout << "ERROR: no name found\n";
			}
			names.push_back(roomName);
			contents.push_back(roomContents);
			exits.push_back(roomExits);

			roomName = "";

		}
		else if (parts[0] == "name")
		{
			roomName = parts[1];
		}
		else if (find(directions.begin(), directions.end(), parts[0]) != directions.end())
		{
			int direction = find(directions.begin(), directions.end(), parts[0]) - directions.begin();
			roomExits[direction] = parts[1];
		}
		else if (parts[0] == "items")
		{
			for (int i=1; i < parts.size(); i++)
			{
				// roomContents->push_back(parts[i]);
				roomContents.push_back(parts[i]);
			}
		}
		else
		{
			// cout << line << "\n";
		}

	}
	roomFile.close();
}

int AttemptMove(int curr, vector<string> & parts, vector<string> & roomNames, vector<string *> exits, vector<string> &
directions)
{

	int direction = find(directions.begin(), directions.end(), parts[0]) - directions.begin();
	cout << direction << "\n";

	int newRoom = curr;
	if (!exits[curr][direction].empty())
	{
		for (int i=0; i < roomNames.size(); i++)
		{
			if (roomNames[i] == exits[curr][direction])
			{

				cout << "You move to the " << exits[curr][direction] << "\n";
				newRoom = i;
			}

		}
	}
	else
	{
		cout << "You can't go that way\n";
	}


	return newRoom;
}

int main()
{
	bool IsActive = true;

	vector<string> DIRECTIONS = {"north", "south", "east", "west", "up", "down"};

	vector<string> inventory;
	vector<string> roomInv;

	vector<string> roomNames;
	vector<vector<string> > roomContents;
	vector<string *> exits;

	InitRooms(roomNames, roomContents, exits, DIRECTIONS);

	string input;

	int currRoom = 0;

	cout << "Welcome to an adventure. There are places you can go: ";
	printVec(roomNames);
	cout << "\n";

	while (IsActive)
	{
		cout << "You are in the " << roomNames[currRoom] << "\n";
		if (roomContents[currRoom].size() > 0)
		{
			cout << "There are some things here: ";
			printVec(roomContents[currRoom]);
			cout << "\n";
		}

		cout << "Exits: \n";
		for (int i=0; i<6; i++)
		{
			if (!exits[currRoom][i].empty())
```

```
                {
                        cout << DIRECTIONS[i] << " to " << exits[currRoom][i] << "\n";
                }
        }

        if (inventory.size() > 0)
        {
                cout << "You have: ";
                printVec(inventory);
                cout << "\n";
        }

        // output the current room
        cout << "->";

        // accept an input
        getline(cin, input);

        vector<string> parts = getWordsFromLine(input);
        if (parts[0] == "quit" || parts[0] == "q")
        {
                IsActive = false;
        }
        else
        {
                currRoom = AttemptMove(currRoom, parts, roomNames, exits, DIRECTIONS);
                ProcessInventoryAction(parts, inventory, roomContents[currRoom]);
        }
    }

    // be sure to clean up memory
    for (int i=0; i<exits.size(); i++)
    {
        delete[] exits[i];
    }


    return 0;
}
```