**BLG 381E**

Advanced Data Structure

2011 Summer School

# Report of Homework 2

Date of Submission : 21.07.2011

Student Name : Ozan Arkan Can

Student Number : 040090573

Instructor : Arzucan Özgür Türkmen

Teaching assistant: Ahmet Aycan Atak

CRN: 30621

# İçindekiler



# Introduction

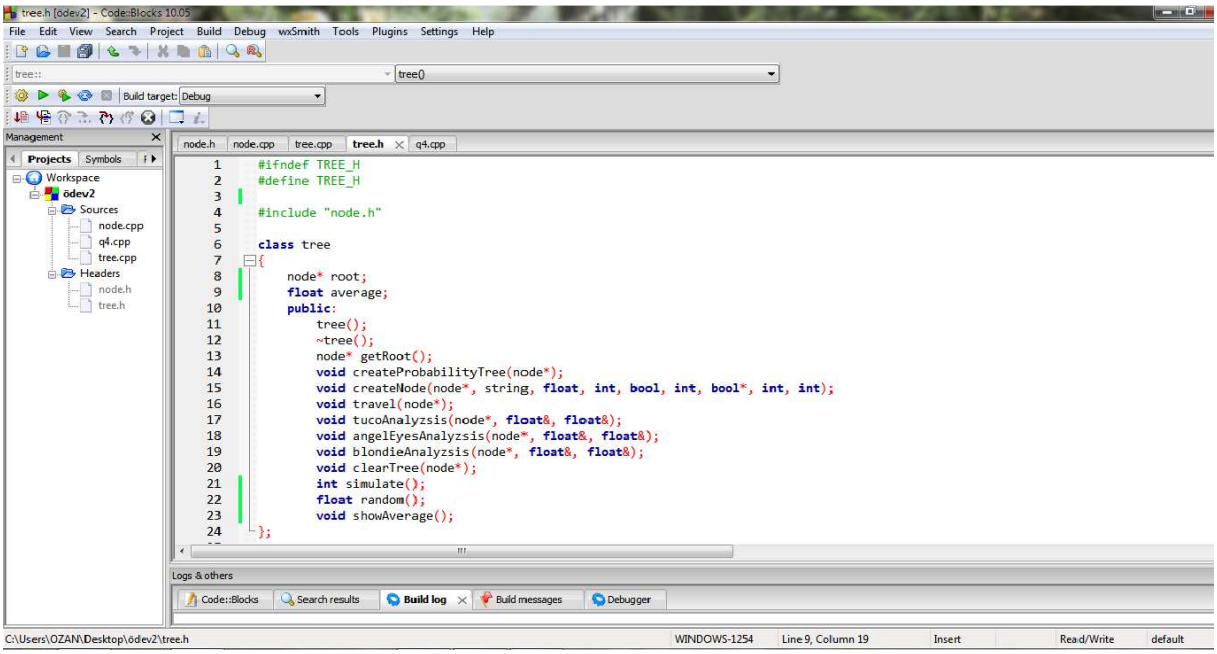
Purpose of this homework is, creating a probability tree for a duel circumstances. Calculating best, average and worst finishing situation and simulating this duel are other jobs that must be done for his homework. Rules of duel are here:

At each turn, they shoot once and they have the same probability to shoot. For example, each of them has the probability value 1/3 to shoot at the beginning of the turn. If Blondie shoots, then he can't shoot until next turn and both Tuco and Angel Eyes have the probability 1/2 to shoot. If Tuco shoots as second, Angel Eyes shoots and turn ends. New turn begins with living ones. Blondie misses with the probability 20%, Angel Eyes misses with the probability 25% and Tuco always misses because his gun is unloaded. Each time they miss, their probability of missing decreases 5% except for Tuco. Blondie always aims to Angel Eyes, Angel Eyes aims to Blondie with probability 85% or he aims to Tuco with probability 15%. Angel Eyes wins the fight if he kills Blondie. Blondie wins the fight if he kills Angel Eyes. Tuco wins the fight if he is still alive.

# Development and Operating Environments

## Ms Windows

The Code::Blocks IDE has been used to write source code, compile and run the code.




## Unix

The source code has been also copied to Unix, then compiled and tested with the GNU C++ Compiler. The following is the commands used:
To compile :

g++ tree.h tree.cpp node.h node.cpp q1.cpp –o q1 (For question1)

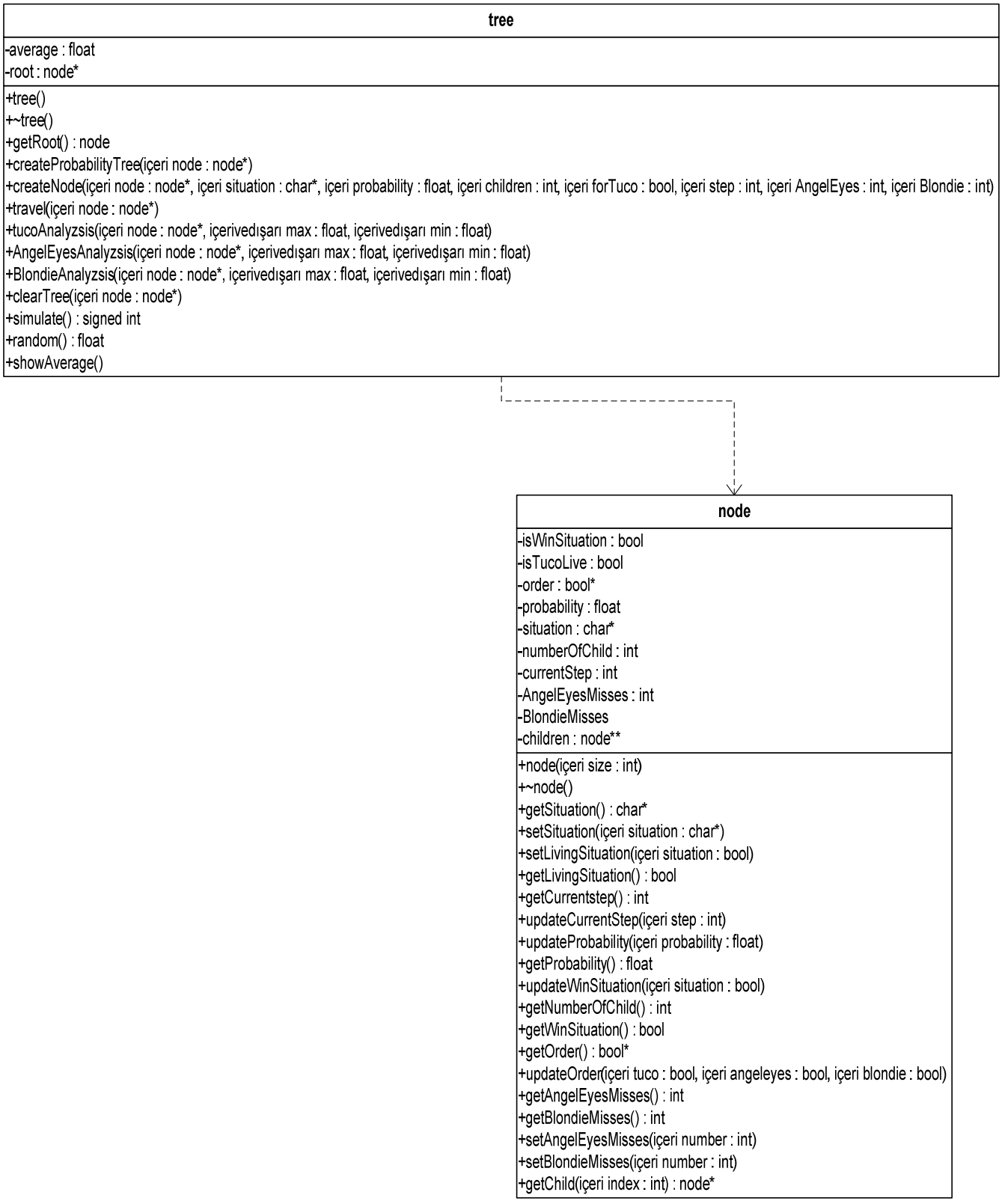
g++ tree.h tree.cpp node.h node.cpp q2.cpp –o q2 (For question2)

g++ tree.h tree.cpp node.h node.cpp q3.cpp –o q3 (For question3)

g++ tree.h tree.cpp node.h node.cpp q4.cpp –o q4 (For question4)

To run: ./q1

# UML Class Diagram



# Q1

CreateProbabilityTree function is created all possible situation in duel.

```
void tree::createProbabilityTree(node* current)
{

    if(!current->getWinSituation())
    {

        string situation = current->getSituation();
        bool* order = current->getOrder();
            if(situation == "root")
            {
                order[0] = true;
                createNode(current->getChild(0), "Tuco shoots", current->getProbability() * 0.333, 1, true, current->getCurrentStep() , order, current->g
                cout << "Hi" << endl;
                order[0] = false; order[1] = true;
                createNode(current->getChild(1),"Angel Eyes shoots", current->getProbability() * 0.333, 2, true, current->getCurrentStep() , order,  curr
                order[1] = false; order[2] = true;
                createNode(current->getChild(2),"Blondie shoots", current->getProbability() * 0.333, 1, true, current->getCurrentStep() , order,  current
            }
            if(situation == "Tuco shoots")
            {
                if(!order[1] && !order[2])
                    createNode(current->getChild(0),"Tuco misses", current->getProbability(), 2, true, current->getCurrentStep() + 1 , order, current->ge
                else
                    createNode(current->getChild(0),"Tuco misses", current->getProbability(), 1, true, current->getCurrentStep() + 1 , order, current->ge
            }
```

# Q2

```
void tree::tucoAnalyzsis(node* current, float& max, float& min)
{
    if(current->getWinSituation())
    {
        if(current->getSituation() == "Blondie dies, Angel Eyes and Tuco win" || current->getSituation() == "Angel Eyes dies, Blondie and Tuco win")
        {
            if(current->getProbability() < min)
                min = current->getProbability();
            if(current->getProbability() > max)
                max = current->getProbability();
        }
    }
    else
    {
        for(int i = 0; i < current->getNumberOfChild(); i++)
            tucoAnalyzsis(current->getChild(i), max, min);
    }
}
```

```
void tree::angelEyesAnalyzsis(node* current, float& max, float& min)
{
    if(current->getWinSituation())
    {
        if(current->getSituation() == "Blondie dies, Angel Eyes and Tuco win" || current->getSituation() == "Blondie dies, Angel Eyes wins")
        {
            if(current->getProbability() < min)
                min = current->getProbability();
            if(current->getProbability() > max)
                max = current->getProbability();
        }
    }
    else
    {
        for(int i = 0; i < current->getNumberOfChild(); i++)
            tucoAnalyzsis(current->getChild(i), max, min);
    }
}
```

```
void tree::blondieAnalyzsis(node* current, float& max, float& min)
{
    if(current->getWinSituation())
    {
        if(current->getSituation() == "Angel Eyes dies, Blondie wins" || current->getSituation() == "Angel Eyes dies, Blondie and Tuco win")
        {
            if(current->getProbability() < min)
                min = current->getProbability();
            if(current->getProbability() > max)
                max = current->getProbability();
        }
    }
    else
    {
        for(int i = 0; i < current->getNumberOfChild(); i++)
            tucoAnalyzsis(current->getChild(i), max, min);
    }
}
```

tucoAnalyzsis , angelEyesAnalyzsis and blondieAnalyzsis functions calculate best and worst winning probabilities for each character by using probability tree that created by q1.

## Q3

```
int tree::simulate()
{
    node* tmp = root;
    while(!tmp->getWinSituation())
    {
        if(tmp->getSituation() != "root")
        {
            cout << tmp->getSituation() << endl;
            cout << "Number of Step: " << tmp->getCurrentStep() << endl;
            cout << "Probability: " << tmp->getProbability() << endl;
            cout << endl;
        }
        if(tmp->getNumberOfChild() == 1)
            tmp = tmp->getChild(0);
        else
        {
            float key = random();
            if(tmp->getSituation() == "root")
            {
                if(key < 0.33)
                    tmp = tmp->getChild(0);
                else
                    if(key < 0.66)
```

simulate function simulates  a duel using probability tree. Starts from root and chooses one child by a random number and probabilities of children until winning situation node.

## Q4

While probability tree is creating , average winning step is also calculated. By calculate average time, following formula is used.

$$average\ step = \sum_{for\ all\ winning\ node} probabiliy\ of\ node * current\ step$$

## Q5

For this question q3 and q4 is used.

## Q6

We can say for one person, he choose one person, he shoots him and he achieve or misses for his shot. If we assume that everyone misses accept last one, number of all situations can be n(number of people)*(n-1)(number of target)*2(misses or achives) = $O(n^2)$

Best case happens when the winning situation occurred first shoot with highest probability. It can be three steps.

Worst case happens when all possible “misses” situations occurred.

## Q7

I have never been watched a western movies, so I have no idea about western movies.

## Conclusion

Values that required to solve could not calculated, because createNode function does not execute well and it causes crash on createProbabilityTree function.

By this homework, creating a probability, simulating situations from a tree is learned.