Introduction to Computer

Homework 06

Due: 2019/06/5

Homework Rules:

Writing homework should be **brought to class** and handed in **before lecture starts**.

If you would late, please bring the writing homework to room BL631 for TA.
(Better to send a mail to TA first in the case that there is no one in room BL631)

As for **programming homework**, you should **upload** it to our course in CEIBA. Uploading deadline would be the coming midnight at 27**:00** .

The file you upload must be a .zip file that contains the following files:
README.txt,
for C++: Agents/b07901XXX.h
for Python: Agents/pyb07901xxx.h and Agents/pyb07901xxx.py

1. Do not submit executable files (.exe) or files for linker(.o, .obj). Files with names in wrong format will not be graded. You must **remove any system functions or windows-only functions,** such as system ("pause"), in your code if you use it.
2. In README.txt file, you need to describe which compiler you choose in this homework and how to compile it (if it is in a “project” form).
3. In your .cpp files, we suggest you write comments as detailed as you can. If the code does not work properly, code with comments can get partial points. It will be good for the TAs to read your code as well as for your future reference and maintenance.

Review Problems (50%)

Chapter 11: Problems 25, 44, 55.
Chapter 12: Problems 13, 17, 31.

Programming Problem (50%)

1. Snake AI

Have you ever play some games like “Snake” with your classmates before? In this homework, we are going to implement an artificial intelligence agent to compete with each other!! Here we **already have a “Snake AI” game** written in C++. All the tanks are controlled by the class “PolicyMaker”. What you need to do is to **inherit it with your own strategy**. Then enjoy having fun with your friends.

What to do?

(1) Build and run the “Snake” project with “Curses” library. **Make sure your environment is well prepared**.
(2) Have some trial, then be more familiar with the rules and game structure.
(3) For C++: Modify the file “b07901xxx.h” into your own version. Implement your agent.
For Python: Modify the file “pyb07901xxx.h” and “pyb07901xxx.py” into your own version. Implement your agent.
(4) Register it in the class “AgentsMgr”. Set your ID for the game and have fun.

How to start? (Project Build and Curses Library)

This game uses the “Curses” library for display controlling. Curses is a terminal control library for the construction of text user interface (TUI) applications. For more detail, you can refer to: http://en.wikipedia.org/wiki/Curses_(programming_library)“Curses” is a cross-platform library, but does not build-in in ANSI C++. You need to download/install it first.

For Windows users, we advise you using the “PDCurses” library (It is the light weight version). And we already pack a **Windows** version PDCurses for you in the homework project file. **Just open the Code::Blocks project and build it**. You need to adjust the size of the command line window to 200*50.Reference:https://www.howtogeek.com/howto/19982/how-to-make-the-windows-command-prompt-wider/

As for **Linux users, you should install the following package.**

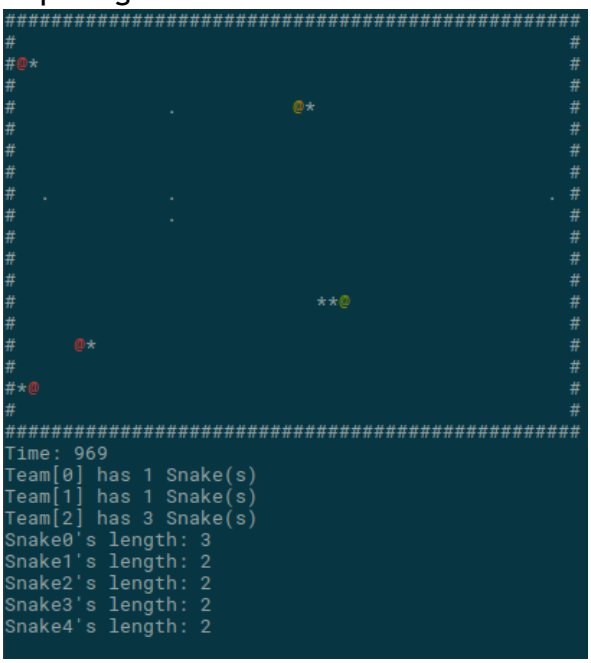
- Debian/Ubuntu: sudo apt-get install libncurses-dev
- RedHat/CentOS: sudo yum install ncurses-devel

As for **Mac users,** curses library is installed by default.

Finally, you can just build the source by typing the command “sh run.sh” in the directory.

First glance at the game

First time you build and the Snake project, you would play it in the "humanGame" mode (see main.cpp). You are controlling (↑↓←→ button) the green snake competing with other snakes. Here is an example:



The game points is calculating by: (body size * 10)

Implement your agent:

For C++ :

Write your AI agent in the file "b07901xxx.h" (rename it as your student ID) :

```cpp
#ifndef __b07901xxx_h__
#define __b07901xxx_h__

//!! TODO 1: modify the ifndef/define protection as your ID like "__b07901xxx_h__"

#include "../PolicyMaker.h"

//!! TODO 2: rename your agent class name as "Agent_b07901xxx" with your own student ID
class Agent_b07901xxx:public PolicyMaker{
public:
    //!! TODO 3: put your student ID for the constructor of PolicyMaker (the base class)
    // you can have argument(s), but all these argument(s) must have their default value
    Agent_b07901xxx():PolicyMaker("b07901xxx"){}

    //{ ===== you can add any member functions and datas here =====
    //but don't use any static data!!!
    Action randMove(){
        int r = rand()%6;
        //2/6%: just go ahead
        //4/6%: 1/6% for each dir
        switch(r){
        case 0:
        case 1:
        case 2:
            return U_Act;
        case 3:
            return D_Act;
        case 4:
            return L_Act;
        case 5:
            return R_Act;
        default:
            return noAct;
        }
    }
    //} ================================================

//!! TODO 4: implement your own actionToDo function here
    virtual Action actionToDo(int arg){
        // view (radius = 3):
        // 00 01 02 03 04
        // 05 06 07 08 09
        // 10 11 me 13 14
        // 15 16 17 18 19
        // 20 21 22 23 24
```

For Python:

First, you should adjust the file “pyb07901xxx.h” (rename it as your student ID):

```h
#ifndef __pyb07901xxx_h__
#define __pyb07901xxx_h__

//!! TODO 1: modify the ifndef/define protection as your ID like "__pyb07901xxx_h__"

#include "../PolicyMaker.h"
#include <fstream>


//!! TODO 2: please change the file name to your ID
const char studentID[100]="pyb07901xxx";

//!! TODO 3: rename your agent class name as "Agent_pyb07901xxx" with your own student ID
class Agent_pyb07901xxx:public PolicyMaker{
public:
//!! TODO 4: put your student ID for the constructor of PolicyMaker (the base class)
// you can have argument(s), but all these argument(s) must have their default value

    Agent_pyb07901xxx():PolicyMaker("pyb07901xxx"){}

    Action getMove(int r){
        //2/6%: just go ahead
```

Write your AI agent in the file “pyb07901xxx.py” (rename it as your student ID) :

```python

# !! TODO 5: implement your own actionToDo function here
def actionToDo(arg):
    # !! Here are some example for python to get view
    # map = f.readline()
    # print("map",map)

    # View = view(map)
    # View.getFood(2)
    # View.getWall(2)

    # view (radius = 1):
    # 00 01 02
    # 03 me 05
    # 06 07 08

    # view (radius = 2):
    # 00 01 02 03 04
    # 05 06 07 08 09
    # 10 11 me 13 14
    # 15 16 17 18 19
    # 20 21 22 23 24
    # Max radius is 3

    # view (radius = 3):
    # 00 01 02 03 04 05 06
    # 07 08 09 10 11 12 13
    # 14 15 16 17 18 19 20
    # 21 22 23 me 25 26 27
    # 28 29 30 31 32 33 34
    # 35 36 37 38 39 40 41
    # 42 43 44 45 46 47 48

    return randMove()
```

Your agent's "actionToDo" function would be called every cycle, you need to return either "U_Act"(press↑), " D_Act"(↓), "L_Act"(←), or "R_Act"(→).
You can get the game information by calling the "getView", "getFoodInView", and "getSnakeInView" functions inherited from "PolicyMaker". Then, register it in the file "AgentsMgr.h" and remember your agent ID (as the order of push_back()).

```
#ifndef __AgentsMgr_h__
#define __AgentsMgr_h__

#include<vector>
#include<string>
#include<cstdio>
//!! TODO 1: put your h/cpp files in "agents" folder
//!! TODO 2: include your b07901xxx.h file here
#include "agents/b07901xxx.h"
#include "agents/pyb07901xxx.h"

// function pionter
typedef PolicyMaker* (*pfNewAgent)(void);

template<class T>
PolicyMaker* fNewAgent(){return new T;}

class AgentsMgr{
public:
    std::vector<pfNewAgent>     pAllNewAgentFunc;
    std::vector<std::string>    agentName;
    int** scores;

    AgentsMgr(){
        //!! TODO 3: add your agent class "Agent_b05901xxx" in a new push_back, so TA can "new" your agent
        // number of snakes should be 4 !!!

        pAllNewAgentFunc.push_back(&fNewAgent<Agent_pyb07901xxx>);// for python agent
        //pAllNewAgentFunc.push_back(&fNewAgent<Agent_b07901xxx>);// for c++ agent

        pAllNewAgentFunc.push_back(&fNewAgent<RandomAgent>);
        pAllNewAgentFunc.push_back(&fNewAgent<RandomAgent>);
        pAllNewAgentFunc.push_back(&fNewAgent<RandomAgent>);

```

Last, change the game mode to "battleAll" in main.cpp.
(↑Just rename all the b07901xxx as your student ID is OK.)

```
int main(){
    Game game;
    // TA: choose one of the following mode to start the game: humanGame,singleGame,battleAll
    // you are the green snake

    //game.humanGame(4,0,0);
        // TA: parameters: int randSeed=4,int aiAgent=0,int viewL=0

    game.battleAll("test.csv",true);
        // TA: parameters: const char* dumpFileName,bool showGame=false
```

[Hint] For printing debugging message, you can make use of (TA's) std::string DebuggingMessage, which would be printed automatically by game every cycle.

Grading:

For basic part (50%), your agent needs to get more points than the "RandomAgent", that is, pointSum("You") >= 5000 over 10 games (Game random seed: 1, 2, 6 ,24, 120, 720, 5040, 362880, 3628800, 39916800). Then you can get the full points.

	A	B	C	D	E
1	0	b07900xxx	RandomAgent	RandomAgent	RandomAgent
2	b07900xxx	7650	0	0	0
3	RandomAgent	0	1270	0	0
4	RandomAgent	0	0	570	0
5	RandomAgent	0	0	0	470

Submitting:

For C++: Just submit the "b07901xxx.h" files (the file name must be "your" student ID).
For Python: Just submit the "pyb07901xxx.h" and "pyb07901xxx.py" files (the file name must be "your" student ID).
Your code must be compatible with our project, and obeys the following rules.

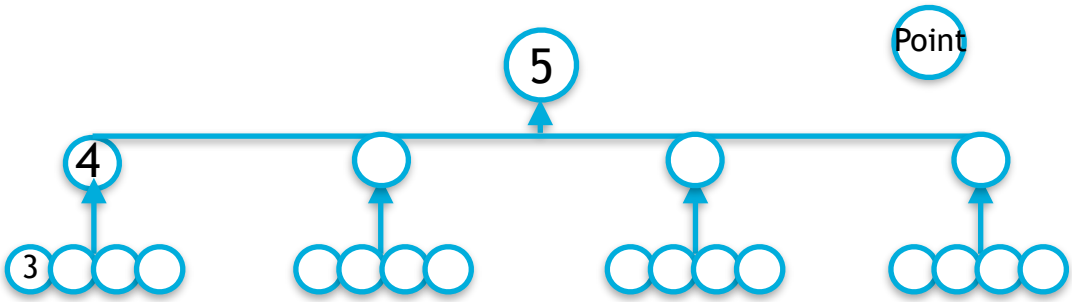
Important rules:

You **MUST follow these coding rules**:

(1) You can **NOT use const_cast** (or any thing like that) to "set" variables or use non-constant member functions of current tanks.
(2) Do **NOT use any global** variable, global function. Just put them in your class.
(3) Also, you can **NOT use static data member**. Memory across different games are not allowed.
(4) You can write more than one agent in "b07901xxx.h", is OK. But **ONLY the class named "Agent_b07901xxx" would be taken for grading**. And other classes must be named as "Agent_b07901xxx_xx…xx" in the case not to conflict with other students.
(5) You can NOT use your own #define or macro function in "b07901xxx.h" file.
(6) Your function should not cost more than 0.1 sec of time, to speed up the game.
(7) Should not use the "NN" library, because we don't want to let the hardware resources to limit the scores of every. If you have any library problems, you can mail to the TA.

2. [Bonus] 5% Competing with the whole class

TA will run the “battleAll” mode with top 16 students’ work. We will choose the top 16th students to the next part, the single elimination tournament. You will receive the bonus points as your rank:



[Hint] You can link each other by putting agents into the same project.