```
/*
    GNU GENERAL PUBLIC LICENSE
    Version 3, 29 June 2007

    This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    You should have received a copy of the GNU General Public License
    along with this program.  If not, see <http://www.gnu.org/licenses/>.

    Permissions beyond the scope of this license may be available at
    http://www.robopartans.com
*/

/*
        VERSION: 1.0.0

        This is the software program writen in NXC for PianoboT - a piano robot
        developed in School of robotics "Robopartans". You could find
        further information, materials and videos of the robot on our site
        http://www.robopartans.com with direct link:
        http://www.robopartans.com/robots/pianobot/
*/

#define LIGHT_TIME 1000
#define P01 190 //degrees to move from position 0 to position 1
#define P12 105 //degrees to move from position 1 to position 2
#define P23 105 //degrees to move from position 2 to position 3
#define POWER 30
#define MAX_POSITION 3 // the maximum position. First position is position 0.
#define PLAMP OUT_A // position lamp
#define SLAMP OUT_C // success lamp
#define MOTOR OUT_B
int transitions[] = {P01, P12, P23};

const int waitForPressTime = 2000;

void errorTone() {
        PlayToneEx(300,300, 100, false);
        Wait(1000);
}

void moveRight(const int currentPosition) {
        switch(currentPosition) {
                case 0: RotateMotor(MOTOR, POWER, P01);
                        break;
                case 1: RotateMotor(MOTOR, POWER, P12);
                        break;
                case 2: RotateMotor(MOTOR, POWER, P23);
                        break;
                default:
                        errorTone();
                        break;
        }
}

void moveLeft(int currentPosition) {
        switch(currentPosition) {
                case 1: RotateMotor(MOTOR, -POWER, P01);
                        break;
```

```
                case 2: RotateMotor(MOTOR, -POWER, P12);
                        break;
                case 3: RotateMotor(MOTOR, -POWER, P23);
                        break;
                default:
                        errorTone();
                        break;
        }
}

void positionLight() {
        OnFwd(PLAMP, 75);
        Wait(1000);
        Off(PLAMP);
}

void success() {
        OnFwd(SLAMP, 75);
        Wait(1000);
        Off(SLAMP);
}

void fail() {
        OnFwd(PLAMP, 75);
        PlayToneEx(300,300, 100, false);
        Wait(3000);
        Off(PLAMP);
}

int moveRandomRight(int currentPosition) {
        int shiftPositions = Random(MAX_POSITION - currentPosition);
        repeat(shiftPositions) {
                moveRight(currentPosition);
                currentPosition++;
                Wait(1000);
        }
        return currentPosition;
}

void moveRightToPosition(const int currentPosition, const int nextPosition) {
        int degreesSum = 0;
        if(currentPosition >=3) {
                errorTone();
                return;
        }
        for(int i=currentPosition; i<nextPosition; i++) {
                degreesSum += transitions[i];
        }
        RotateMotor(MOTOR, POWER, degreesSum);
}

void moveLeftToPosition(const int currentPosition, const int nextPosition) {
        int degreesSum = 0;
        if(currentPosition<=0) {
                errorTone();
                return;
        }
        for(int i=currentPosition; i>nextPosition; i--) {
                degreesSum += transitions[i-1];
        }
        RotateMotor(MOTOR, -POWER, degreesSum);
}

int moveToPosition(const int currentPosition, const int nextPosition) {
        if(currentPosition<nextPosition) {
                moveRightToPosition(currentPosition, nextPosition);
        } else if(currentPosition>nextPosition) {
```

```
                moveLeftToPosition(currentPosition, nextPosition);
        }
}
int showLevel(int level[]) {
        int currentPosition =0;
        for(int i =0;i<ArrayLen(level); i++) {
                moveToPosition(currentPosition, level[i]);
                positionLight();
                currentPosition = level[i];
        }
        return currentPosition;
}
/*
Return the number pressed button or -1 if a button is not pressed. First button is with nubmer 0.
*/
bool expectButton(int expected) {
        int startTime = CurrentTick();
        int endTime = CurrentTick();
        bool pressed = false;
        while(endTime - startTime <waitForPressTime) {
                endTime = CurrentTick();
                if(SENSOR_1 ==1 && expected == 0) {
                        return true;
                }
                else if(SENSOR_2 == 1 && expected ==1) {
                        return true;
                }
                else if(SENSOR_3 == 1 && expected ==2) {
                        return true;
                } else if(SENSOR_4 == 1 && expected == 3) {
                        return true;
                }
        }
        return false;
}

bool expectLevel(int level[]) {
        bool pressed = false;
        for(int i =0; i<ArrayLen(level); i++) {
                pressed = expectButton(level[i]);
                if(pressed == false)
                        break;
        }
        return pressed;
}

void playLevel(int level[]) {
        showLevel(level);
        moveLeftToPosition(level[ArrayLen(level) -1], 0);
        bool passed = expectLevel(level);
        if(passed) {
                success();
                Wait(2000);
        } else {
                fail();
                Stop(true);
        }

}
task main() {

        SetSensor(IN_1, SENSOR_TOUCH);
        SetSensor(IN_2, SENSOR_TOUCH);
        SetSensor(IN_3, SENSOR_TOUCH);
        SetSensor(IN_4, SENSOR_TOUCH);
        ClearSensor(IN_1);
        ClearSensor(IN_2);
```

```
        ClearSensor(IN_3);
        ClearSensor(IN_4);
        int level_1[] = {1};
        int level0[] = {1,3};
        int level1[] = {1,3,0};
        int level2[] = {1,3,0,2};
        int level3[] = {1,3,0,2,3};
        int level4[] = {1,3,0,2,3,1};
        int level5[] = {1,3,0,2,3,1,0};
        int level6[] = {1,3,0,2,3,1,0,2};
        int level7[] = {1,3,0,2,3,1,0,2,3};
        int level8[] = {1,3,0,2,3,1,0,2,3,0};

        playLevel(level_1);
        playLevel(level0);
        playLevel(level1);
        playLevel(level2);
        playLevel(level3);
        playLevel(level4);
        playLevel(level5);
        playLevel(level6);
        playLevel(level7);
        playLevel(level8);
        repeat(5) {
                success();
                Wait(400);
        }
}
```