出題日 2019 年 6 月 14 日（金），6 月 28 日（金）まで各グループで指定された方法で提出

グループ 1：甲斐（s01060@cc.seikei.ac.jp）　グループ 2：世木（s02415@cc.seikei.ac.jp）
グループ 3：松田（s02767@cc.seikei.ac.jp）　グループ 4：呉　（s02612@cc.seikei.ac.jp）
グループ 5：坂本（s02684@cc.seikei.ac.jp）

レポート 4

2 次元の迷路をキューとスタックのデータ構造を利用して解いてみよう．地図情報は $n \times n$ の 2 次元配列で与えられ，スタートを点 $(0,0)$，ゴールを点 $(n-1, n-1)$ とする．配列の各セルに「-1」（壁）と「0」（通過可能）の初期値が入っている．問題 1–3 に回答せよ．問題 3 は発展問題とする．

問題 1（必須）

演習 9 の問題 2 で利用した幅優先探索のアイデアを生かして，2 次元座標を格納できるキューで迷路を解く．Point，Node と Queue クラスの定義は演習 9 の問題 1 で作成したものを利用してよい．main 関数などの一部の関数が書かれたファイル repo4-skel.cpp（Code 10）と問題例ファイル（maze1.txt，maze2.txt，maze3.txt）を `/share/material/advpro/2019/` に用意してあるので使用して良い．cpp ファイルの関数 solveByQueue を補完し，以下の結果になるように完成させなさい．

Code 1　maze1 の実験結果

```
comsv\% ./repo-4
問題例ファイルを入力してください：maze1.txt
■■■■■■
■  ×   ■
■     ×■
■×  ××■
■  ×   ■
■■■■■■
Queue：ゴールまで到達不可能
```

Code 2　maze2 の実験結果

```
comsv\% ./repo-4
問題例ファイルを入力してください：maze2.txt
■■■■■■■
■    ××  ■
■  ×     ■
■     ×  ■
■××  ×  ■
■     ×  ■
■■■■■■■
Queue：ゴールまで到達可能
```

Code 3　maze3 の実験結果

```
comsv\% ./repo-4
問題例ファイルを入力してください：maze3.txt
■■■■■■■■■
■  ×       ■
■     ×  ×■
■×  ××  ×■
■          ■
■×  ×  ×  ■
■    ×  ×  ■
■■■■■■■■■
Queue：ゴールまで到達可能
```

ヒント（探索手順）：

1) 空のキューを用意する．
2) スタート点をキューに追加する．
3) キューから点 1 つを取り出し，その点から隣接点の探索を行い，まだ未探索ならキューに追加する．その操作をキューが空になるまで繰り返す．
4) ゴール点の状態を確認し，到達可能性を返す．

問題２（必須）

２次元座標を格納できる Stack を利用して，スタートからゴールまでの経路出力と地図描画ができるように printPath 関数を補完し，以下の結果になるように完成させなさい．Stack クラスの定義は演習 8 の問題 1 で作成したものから修正してよい．

Code 4　maze1 の実験結果

```
comsv\% ./repo-4
問題例ファイルを入力してください：maze1.txt
■■■■■■
■  ×   ■
■     ×■
■×  ××■
■  ×   ■
■■■■■■
Queue：ゴールまで到達不可能
```

Code 5　maze2 の実験結果

```
comsv\% ./repo-4
問題例ファイルを入力してください：maze2.txt
■■■■■■■
■    ××  ■
■  ×     ■
■     ×  ■
■××  ×  ■
■     ×  ■
■■■■■■■
Queue：ゴールまで到達可能
Path：(0, 0) (1, 0) (2, 0) (2, 1) (2, 2)
    (1, 2) (1, 3) (1, 4) (2, 4) (3, 4)
    (4, 4)
■■■■■■■
■●  ××  ■
■●×●●●■
■●●●×●■
■××  ×●■
■     ×●■
■■■■■■■
```

Code 6　maze3 の実験結果

```
comsv\% ./repo-4
問題例ファイルを入力してください：maze3.txt
■■■■■■■■
■  ×      ■
■     ×  ×■
■×  ××  ×■
■         ■
■×  ×  ×  ■
■    ×  ×  ■
■■■■■■■■
Queue：ゴールまで到達可能
Path：(0, 0) (1, 0) (1, 1) (2, 1) (3, 1)
    (3, 2) (3, 3) (3, 4) (3, 5) (4, 5)
    (5, 5)
■■■■■■■■
■●×      ■
■●●  ×  ×■
■×●××  ×■
■  ●●●●●■
■×  ×  ×●■
■    ×  ×●■
■■■■■■■■
```

ヒント：
経路探索する際にどの隣接セルから来たかの情報を地図配列に保存すると，ゴールからスタートまでの経路が取得できる．その経路は逆順になるので，Stack を利用し，スタートからゴールまでの経路を出力する．また，経路情報を含める地図を描画するには，用意してある printMap 関数を利用してよい．

問題 3（発展）

キューの代わりに，２次元座標を格納できるスタックを利用して迷路を解く．Stack では，最後に追加した要素が，次で使われるので，見つかった順に深堀していくことができる．すなわち，ある地点から 4 方向（上下左右）を探索するときに，最後に追加した探索可能な方向におけるすべて

の経路探索が終わってから，別の方向を試すことになる．関数 solveByStack を補完し，以下の結果になるように完成させなさい．（maze3 について，Stack で解いた結果は探索順序によって示した結果と異なる場合がある．正しい経路となれば，問題ない．）

Code 7　maze1 の実験結果

```
comsv\% ./repo-4
問題例ファイルを入力してください：maze1.txt
■■■■■■
■  ×   ■
■     ×■
■×  ××■
■  ×   ■
■■■■■■
Queue：ゴールまで到達不可能
```

Code 8　maze2 の実験結果

```
comsv\% ./repo-4
問題例ファイルを入力してください：maze2.txt
■■■■■■■
■    ××  ■
■  ×     ■
■     ×  ■
■××  ×  ■
■     ×  ■
■■■■■■■
Queue：ゴールまで到達可能
Path：(0, 0) (1, 0) (2, 0) (2, 1) (2, 2) (1,
    2) (1, 3) (1, 4) (2, 4) (3, 4) (4, 4)
■■■■■■■
■●  ××  ■
■●×●●●■
■●●●×●■
■××  ×●■
■     ×●■
■■■■■■■
Stack：ゴールまで到達可能
Path：(0, 0) (1, 0) (2, 0) (2, 1) (2, 2) (1,
    2) (1, 3) (1, 4) (2, 4) (3, 4) (4, 4)
■■■■■■■
■●  ××  ■
■●×●●●■
■●●●×●■
■××  ×●■
■     ×●■
■■■■■■■
```

Code 9　maze3 の実験結果

```
comsv\% ./repo-4
問題例ファイルを入力してください：maze3.txt
■■■■■■■■
■  ×      ■
■     ×  ×■
■×  ××  ×■
■         ■
■×  ×  ×  ■
■   ×  ×  ■
■■■■■■■■
Queue：ゴールまで到達可能
Path：(0, 0) (1, 0) (1, 1) (2, 1) (3,
    1) (3, 2) (3, 3) (3, 4) (3, 5)
    (4, 5) (5, 5)
■■■■■■■■
■●×      ■
■●●  ×  ×■
■×●××  ×■
■  ●●●●●■
■×  ×  ×●■
■   ×  ×●■
■■■■■■■■
Stack：ゴールまで到達可能
Path：(0, 0) (1, 0) (1, 1) (1, 2) (0,
    2) (0, 3) (0, 4) (1, 4) (2, 4)
    (3, 4) (3, 5) (4, 5) (5, 5)
■■■■■■■■
■●×●●●  ■
■●●●×●×■
■×  ××●×■
■      ●●■
■×  ×  ×●■
■   ×  ×●■
■■■■■■■■
```

Queue を利用するときの幅優先探索と Stack を利用するときの深さ優先探索を比較し，それぞれの特徴を分析してみよう．また，今回の迷路問題を解くにはどのデータ構造を利用した方がよいかを議論し，理由を述べてみよう．

補完する cpp ファイル

- 問題 1 ：solveByQueue 関数を補完しなさい.
- 問題 2 ：printPath 関数を補完しなさい.
- 問題 3 ：main 関数にあるコメントアウトした部分を有効にし，solveByStack 関数を定義しなさい.

Code 10 report4-skel.cpp

```
#include <iostream>
#include <fstream>
#include <cstdlib>
using namespace std;
// Node, Queue と Stack クラスはここに定義
// または include してもよい

// 迷路に設置するセルの属性
enum{
    WALL = -1,      // 壁
    EMPTY = 0,      // 通路
    ROUTE = 1,      // ゴールまでの道
    LEFT = 1001,    // 左のセルからきた（探索途中用）
    RIGHT,          // 右のセルからきた（探索途中用）
    UP,             // 上のセルからきた（探索途中用）
    DOWN,           // 下のセルからきた（探索途中用）
    START           // 出発セル（探索途中用）
};

bool solveByQueue(int** arr, const int n){
    // 空のキューを用意する
    Queue q;

    // 始点(0,0)をキューに追加する
    q.enqueue(Point(0, 0));
    arr[0][0] = START;

    // [問題1]幅優先探索を行う（キューを利用してコードを補完する）

    // ゴールセルの状態を確認する
    if(arr[n-1][n-1] == EMPTY)
        return false;
    return true;
}

// Map を初期化する
void initMap(int** arr, const int n){
```

```
    for(int i=0; i<n; i++)
        for(int j=0; j<n; j++)
            if(arr[i][j] != WALL)
                arr[i][j] = EMPTY;
}

// Map をプリントする
void printMap(int** arr, const int n){
    for(int i=-1; i<=n; i++){
        for(int j=-1; j<=n; j++){
            if(i == -1 || i == n || j == -1 || j == n)
                cout << "■";
            else if(arr[i][j] == WALL)
                cout << "×";
            else if(arr[i][j] == ROUTE)
                cout << "●";
            else
                cout << " ";
        }
        cout << endl;
    }
}

void printPath(int** arr, const int n){
    // [問題２] スタックを利用して，ゴールまでの経路と地図を描画する
    // 描画する際に，printMap 関数を利用してよい
}

int main(){
    int n;
    int** ins;

    // 問題入力の読み込み
    char fName[100];
    cout << "問題例ファイルを入力してください：";
    cin >> fName;
    ifstream fin(fName, ios::in);
    if( !fin ){
        cout << "Error: ファイル（" << fName << "）が開けません." << endl;
        exit(1);
    }
    fin >> n;
    ins = new int*[n];
    for(int i=0; i<n; i++){
        ins[i] = new int[n];
        for(int j=0; j<n; j++)
            fin >> ins[i][j];
    }
```

```
    // 入力地図の描画
    printMap(ins, n);

    // キューを利用する迷路探索
   bool rst = solveByQueue(ins, n);
   if(rst){
        cout << "Queue: ゴールまで到達可能" << endl;
     printPath(ins, n);
    }
   else{
     cout << "Queue: ゴールまで到達不可能" << endl;
    }

    /*[問題３] に関係するコード
    // 入力地図の初期化
    initMap(ins, n);

    // スタックを利用する迷路探索
   rst = solveByStack(ins, n);
   if(rst){
        cout << "Stack: ゴールまで到達可能" << endl;
     printPath(ins, n);
    }
   else{
     cout << "Stack: ゴールまで到達不可能" << endl;
    }
    */

    for(int i=0; i<n; i++)
        delete[] ins[i];
    delete[] ins;
    return 0;
}
```