Lab10

Evaluation criteria

Category	Evaluation	
p10	100	
Total	100	

- *Use GCC **11** version*
- *No score will be given if the gcc version is different.*

Lab 10 Maze

- The deadline for lab10 submission is May 17 at 11:59 PM.
- lab10 - init, union_, find, createMaze, printMaze, freeMaze
- Folder name : lab10
- code name: p10.c
- Each code will be tested by 5 different input files.
- 20 score for each input, if you don’t get the answer you get 0 score.

Lab 10 Maze

void init(DisjointSets *sets, PrintDisjointSets *maze, int n)

- Initialize all cells to sets and maze.

void union_(DisjointSets *sets, int i, int j)

- Union two sets.

int find(DisjointSets *sets, int x)

- Find the set including the number and return the representative member of the set.

Lab 10 Maze

void createMaze(DisjointSets *sets, PrintDisjointSets *maze, int n)

- Generate a maze that includes a path from Start position to End position **WITHOUT** any cycles.
- You can generate such a maze by randomly choosing a cell and direction.
- Use Union-Find ADT.
- For random number generation, use the library functions.

void printMaze(PrintDisjointSets *maze, int n)

- Print the resulting maze.

void freeMaze(DisjointSets *sets, PrintDisjointSets *maze)

- Free memory of the maze.

Lab 10 Maze

- Main

```c
int main(int argc, char* argv[]){
    int num;
    FILE *fi = fopen(argv[1], "r");
    fscanf(fi, "%d", &num);
    fclose(fi);

    DisjointSets *sets;
    PrintDisjointSets *maze;

    sets = (DisjointSets*)malloc(sizeof(DisjointSets));
    maze = (PrintDisjointSets*)malloc(sizeof(PrintDisjointSets));

    init(sets, maze, num);

    createMaze(sets, maze, num);

    printMaze(maze, num);

    freeMaze(sets, maze);

    return 0;
}
```

Lab 10 Maze

- Structure

```c
typedef struct _DisjointSet{
    int size;
    int *ptr_arr; // parent
} DisjointSets;
```

```c
typedef struct _PrintDisjointSet{
    int size;
    int *ptr_arr; // wall
} PrintDisjointSets;
```

- Function

```c
void init(DisjointSets *sets, PrintDisjointSets* maze, int n);
int find(DisjointSets *sets, int x);
void union_(DisjointSets *sets, int i, int j);
void createMaze(DisjointSets *sets, PrintDisjointSets *maze, int n);
void printMaze(PrintDisjointSets *maze, int n);
void freeMaze(DisjointSets *sets, PrintDisjointSets *maze);
```

Lab 10 Maze

- init

start

1	2	3	4	5	6
7	8	9	10	11	12
13	14	15	16	17	18
19	20	21	22	23	24
25	26	27	28	29	30
31	32	33	34	35	36

end

- num = 6

- Variable
  - sets: means the number between the walls
  - maze: means the wall (1: yes, 0: no)
  - num = row = column
  - all sets->ptr_arr are 0 (meaning root)
  - all maze->ptr_arr are 1 (meaning wall)
    - Except entrance and exit

Lab 10 Maze

- init - ex) num=6
  - sets

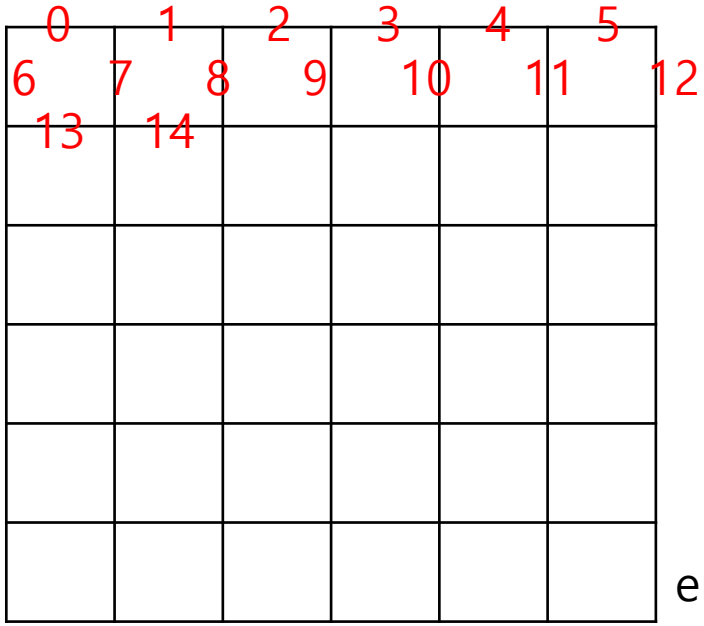
start

1	2	3	4	5	6
7	8	9	10	11	12
13	14	15	16	17	18
19	20	21	22	23	24
25	26	27	28	29	30
31	32	33	34	35	36

end

  - maze_print

start



end

wall8 : cell2, cell3

wall14 : cell2, cell8

For 8, 2 = 8 / (6+6+1) * 6 + 8 % (6+6+1) – 6, 3 = 2 + 1

For 14, 2 = (14 / (6+6+1) – 1) * 6 + 14 % (6+6+1) + 1, 8 = 2 + 6

Lab 10 Maze

- createMaze

1	2	3	4	5	6
7	8	9	10	11	12
13	14	15	16	17	18
19	20	21	22	23	24
25	26	27	28	29	30
31	32	33	34	35	36

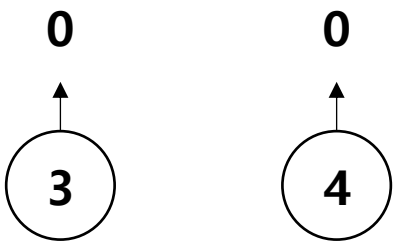
1. select wall
2. cell3, cell4
3. compare the root of cell3 and the root of cell4
4. if the roots are different, merge two cells and remove the wall
5. repeat 1 to 4 until first cell and last cell are in the same set

Lab 10 Maze

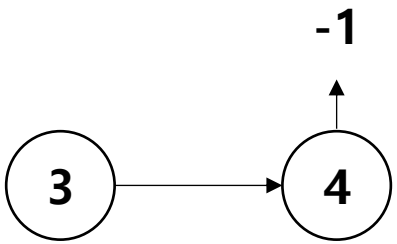
- createMaze

1. select wall: random number
2. root cell: find function
3. merge two cells: union_ function
4. remove the wall: 1 to 0 in maze->ptr_arr

- Random number
  - srand(), rand() - <stdlib.h>
  - time() - <time.h>
  - srand(time(NULL)) //generate seed
  - rand()%10 // 0~9
- find function
  - search until finding root
- union_ function
  - one cell's connect to another cell



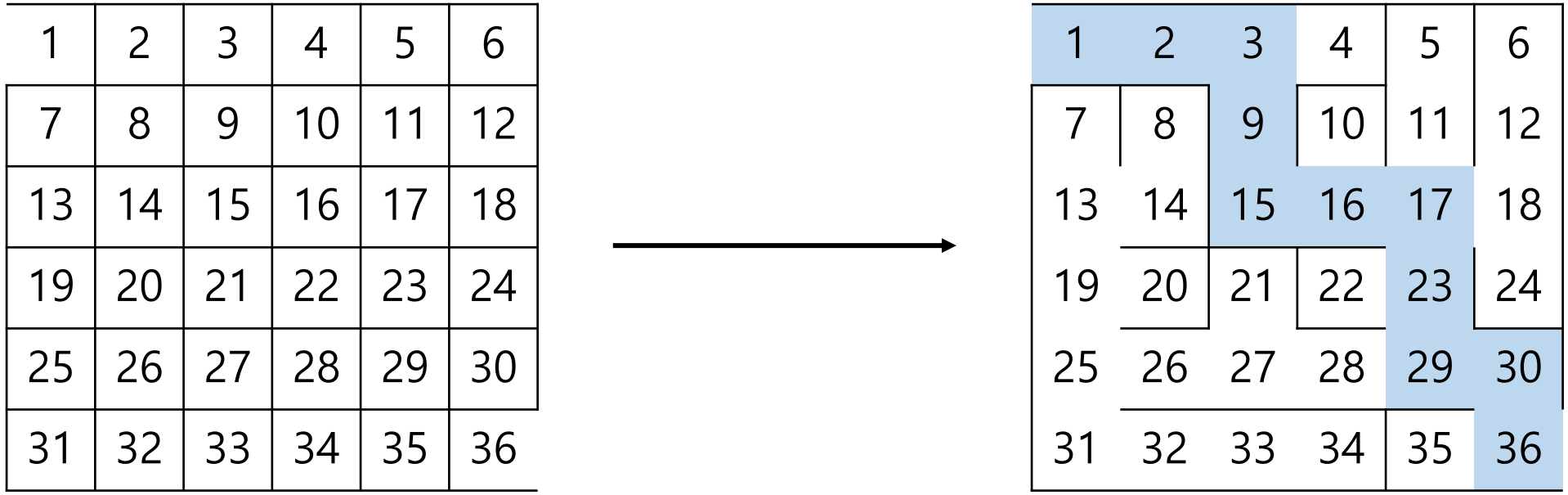
find 3 and 4



connect 3 and 4

Lab 10 Maze

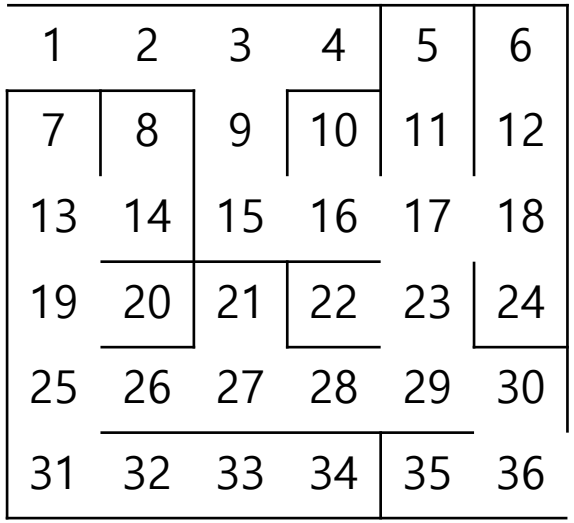
- createMaze

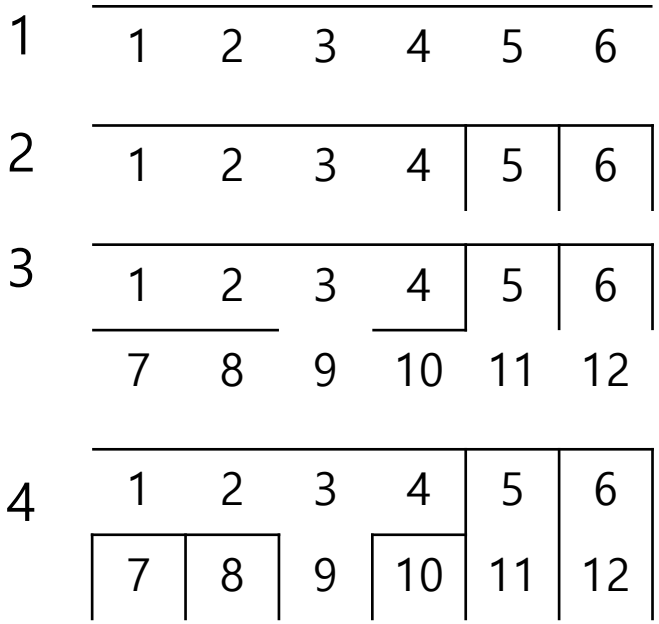


- No cycles
- First cell and last cell are in the same set

Lab 10 Maze

- printMaze

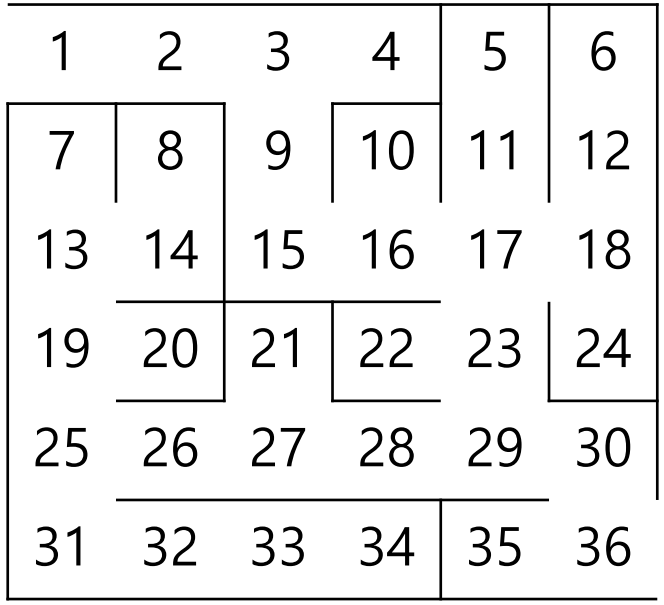




-
-
-

Lab 10 Maze

- printMaze



Open entrance and exit (no walls)

- freeMaze
  - free memory

Lab 10 Maze

- program name : p10.c
- input : an integer in a file.

```
6
~
~
~
~
~
```

- output : the corresponding result in the standard output.

```
root@094d6e320e1f:/home/2021_DS/temp# ./p11 input.txt
 - - - - - -
       | | |
 - -   -
| | | | | | |

|   |       |
   - - -
|   | |   | |
   -   -   -
|           |
   - - - -
|       |
 - - - - - -
```