# DS HW4

- 마감일: (**1분반**) **11월 2일(월)** 자정까지 (**2분반**) **11월 5일(목)** 자정까지
- 제출방법:

```
1분반: submit pem_ta hw4a
2분반: submit pem_ta hw4b
```

※ 제출 시 실행파일을 지우고 제출합니다. (지우기 명령어 : `rm 파일명`)

※ 제출파일 : `hw4.cpp, maze.cpp, maze.in, maze.in2, makefile, .tex, .pdf`

● **주의! 지정한 파일명과 다를 경우 0점 처리합니다.**

1. 아래와 같이 파일을 작성하시오.

   (a) 다음과 같은 `makefile`을 작성하라.

```
cat makefile

hw4:  hw4.o maze.o
  g++ -o hw4 hw4.o maze.o
```

   (b) 12 by 15 짜리 maze를 저장한 `maze.in`을 만드시오(see fig3.11).

```
cat maze.in

12  15
0 1 0 0 0 1 1 0 0 0 1 1 1 1 1
1 0 0 0 1 1 0 1 1 1 0 0 1 1 1
0 1 1 0 0 0 0 1 1 1 1 0 0 1 1
1 1 0 1 1 1 1 0 1 1 0 1 1 0 0
1 1 0 1 0 0 1 0 1 1 1 1 1 1 1
0 0 1 1 0 1 1 1 0 1 0 0 1 0 1
0 0 1 1 0 1 1 1 0 1 0 0 1 0 1
0 1 1 1 1 0 0 1 1 1 1 1 1 1 1
0 0 1 1 0 1 1 0 1 1 1 1 1 0 1
1 1 0 0 0 1 1 0 1 1 0 0 0 0 0
0 0 1 1 1 1 1 1 0 0 0 1 1 1 1 0
0 1 0 0 1 1 1 1 1 1 0 1 1 1 1 0
```

(c) 9 by 9 짜리 maze를 저장한 `maze.in2` 도 만드시오

```
cat maze.in2

9 9
0 0 0 0 0 0 0 0 1
1 1 1 1 1 1 1 1 0
1 0 0 0 0 0 0 0 1
0 1 1 1 1 1 1 1 1
1 0 0 0 0 0 0 0 1
1 1 1 1 1 1 1 1 0
1 0 0 0 0 0 0 0 1
0 1 1 1 1 1 1 1 1
1 0 0 0 0 0 0 0 0
```

2. `hw4.cpp`와 `maze.cpp`를 작성하자.(헤더파일 없음)

   (a) 출력 형식: 다음 같이 동작하는 프로그램을 작성하려 한다. 자료 `maze.in2`에 대해서도 동작하나 확인하라.

```
make hw4
hw4 maze.in

For maze datafile (maze.in)
 -> (1,1) -> (2,2) -> (1,3) -> (1,4) -> (1,5)
 -> (2,4) -> (3,5) -> (3,4) -> (4,3) -> (5,3)
 -> (6,2) -> (7,2) -> (8,1) -> (9,2) -> (10,3)
 -> (10,4) -> (9,5) -> (8,6) -> (8,7) -> (9,8)
 -> (10,8) -> (11,9) -> (11,10) -> (10,11) -> (10,12)
 -> (10,13) -> (9,14) -> (10,15) -> (11,15) -> (12,15)
```

   (b) main 프로그램(`hw4.cpp`)은 다음과 같다.

```
#include <iostream>
#include <fstream>
#include <stdlib.h>
using namespace std;

void getdata(istream&, int&, int&);
void Path(int, int);

int main(int argc, char* argv[])
{
   int m, p; // m by p maze
   if (argc == 1)
      cerr << "Usage: " << argv[0] << " maze_data_file" << endl;
   else {
      ifstream is(argv[1]);
      if (!is) { cerr << argv[1] << " does not exist\n"; exit(1); }
      cout << "For maze datafile (" << argv[1] << ")\n";
```

```
      getdata(is, m, p); is.close();
      Path(m, p);
   }
}
```

(c) 다음 프로그램을 `maze.cpp`로 저장하라.

```
#include <iostream>
#include <stack>
using namespace std;
const int MAXSIZE=100; // up to 100 by 100 maze allowed
bool maze[MAXSIZE+2][MAXSIZE+2];
bool mark[MAXSIZE+1][MAXSIZE+1] = {0};

enum directions { N, NE, E, SE, S, SW, W, NW };
struct offsets
{
   int a, b;
} mov[8] = { /*see fig.3.13 like enum directions*/ };
struct Items {
   Items(int xx=0, int yy=0, int dd=0): x(xx), y(yy), dir(dd) {}
   int x, y, dir;
};

template <class T>
ostream& operator<< (ostream& os, stack<T>& s) {
       // 스택의 내용을 역순으로 출력
       // 구현방법=내용을 하나씩 꺼내 다른 임시 스택에 넣어 저장한 후,
       // 최종적으로 그 임시 스택에서 하나씩 꺼내 출력하면 됨
   return os;
}
ostream& operator<<(ostream& os, Items& item)
{
   // 5개의 Items가 출력될 때마다 줄바꾸기위해
   static int count = 0;
   os << "(" << item.x << "," << item.y << ")";
   count++;
   if ((count % 5) == 0) cout << endl;
   return os;
}


void Path(const int m, const int p)
{

  /* 구현은 책과 동일하다. 단 최종적인 경로의 출력은 다음과 같이 한다.
    cout << stack;
    temp.x = i; temp.y = j; cout << " -> " << temp;
    temp.x = m; temp.y = p; cout << " -> " << temp << endl;
  */

}
void getdata(istream& is, int& m, int & p)
{ // 자료화일을 읽어들여 maze에 저장한다.
   is >> m >> p;
```

```
   for (int i = 0; i < m+2; i++) { maze[i][0] = 1; maze[i][p+1] = 1; }
   for (int j = 1; j <= p; j++) { maze[0][j] = 1; maze[m+1][j] = 1; }
   for (int i = 1; i <= m; i++)
      for (int j = 1; j <= p; j++)
         is >> maze[i][j];
}
```

(d) 다음과 같이 경로 출력 후 방문한 노드 수를 출력하도록 `maze.cpp`를 수정하시오. 방법은 새로 노드를 mark할 때마다 갯수를 세어 Path에서 return직전 그 값을 출력한다.

```
make hw4
hw4 maze.in

For maze datafile (maze.in)
 -> (1,1) -> (2,2) -> (1,3) -> (1,4) -> (1,5)
 -> (2,4) -> (3,5) -> (3,4) -> (4,3) -> (5,3)
 -> (6,2) -> (7,2) -> (8,1) -> (9,2) -> (10,3)
 -> (10,4) -> (9,5) -> (8,6) -> (8,7) -> (9,8)
 -> (10,8) -> (11,9) -> (11,10) -> (10,11) -> (10,12)
 -> (10,13) -> (9,14) -> (10,15) -> (11,15) -> (12,15)

#nodes visited = 48 out of 180
```

3. 보고서 작성
   - stack의 작동방식 설명
   - 경로를 찾는 알고리즘과 stack이 어떻게 사용되었는지 설명
   - 과제 구현 시 어려웠던 점 서술
4. 질문사항

● 조교 이메일(ntommy11@naver.com)로 질문.