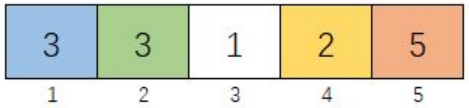
Announcements

- Coming soon: TA video for homework help. This is to provide a starting point/basic idea/hint for homework if you have difficulty.
  - Will be published in classroom
  - We still have TA office hours as usual
  - Try to work on homework without watching video. But if you have no clue, watch the TA video before heading to TA office hour

Strange elevator (P1642)

https://xjoi.net/contest/4435



step 1

step 2

step 3

Sample input:

5 1 5

3 3 1 2 5

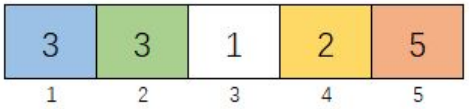
Sample output:

3

Strange elevator

Use BFS

- Input data
- Push the start to the queue
- Add all reachable floors to queue
- Until it reaches the destination or fails
- ps：Watch out for out of bounds



XJOI problem:1642

Strange elevator

```
4  int level[201], visited[201], step[201];
5  int dx[2] = {1, -1};
6
```

```
19  queue<int> q;
20  q.push(start);
21  visited[start] = 1;
22  while(!q.empty()) {
23      int x = q.front();
24      q.pop(); // remove front
25
26      for( i = 0; i < 2; i++) {
27          int nx = x + dx[i]*level[x]; // calc next level
28
29          if (nx >= 0 && nx < n && visited[nx] == 0) { // can move
30              visited[nx] = 1;
31              step[nx] = step[x] + 1; // increase step
32              if (nx == end) {
33                  cout << step[nx] << endl;
34                  return 0;
35              }
36              q.push(nx);
37          }
38      }
39  }
```

Sea Battle (P9431)

In the game, some ships are placed on a square table, and each ship cannot touch other ships. In this problem, we only consider boats to be rectangles, all boats are rectangles composed of "*". Write a program to find the total number of ships placed in the game.

input sample

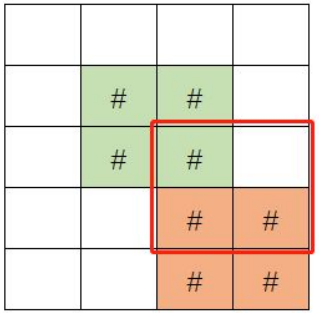
```
6 8
.....#.#
##.....#
##.....#
.......#
#......#
 #..#...#
```

output sample

```
There are 5 ships.
```

Sea Battle (P9431)

- As can be seen from the figure, the five ships are five connected areas



- This is the case of mutual contact, please observe its characteristics

- How do we check the shape is rectangle?

Sea Battle (P9431)

```
    for(int i = 0; i < n; i++)
        for(int j = 0; j < m; j++) {
            if (maze[i][j] == '#') { // found new ship
                maze[i][j] = '.';  // mark visited
                now_count++; // ship number
                int minx = i, miny = j, maxx = i, maxy = j, num = 1; // define
corners
                q.push(PNT(i, j));

                while(!q.empty()) {
                    PNT pt = q.front();
                    q.pop();
                    for(int k = 0; k < 4; k++) {
                        int nx = pt.x + dx[k];
                        int ny = pt.y + dy[k];

                        if (nx >= 0 && nx < n && ny >= 0 && ny < m
                            && maze[nx][ny] == '#' ) { // can move
                                minx = min(minx, nx); // search corners
                                miny = min(miny, ny);
                                maxx = max(maxx, nx);
                                maxy = max(maxy, ny);
                                num++;
                                maze[nx][ny] = '.'; // mark visited
                                q.push(PNT(nx, ny));
                        }
                    }
                }

                // check if rectangle
                int expect = (maxx-minx+1) * (maxy-miny+1);
                if (expect != num) {
                    printf("Bad placement.\n");
                    return 0;
                }
            }

```

Minimum turns (P8121)

What's given?

- N, M: size of the maze matrix
- NxM matrix: 0 path; 1 wall
- start and end points: sx, sy, ex, ey

What's asked?

- the minimum number of turns taken from the start port to the end point.

What's your strategy?

- Try all possible paths.
- Count how many turns happened.
- How to count number of turns? How do we know there is a turn?

If the new direction is different, it's a turn.

```
1000010
0010100
0000101
0110000
0000110
```

Minimum turns (P8121)

What data structure we need?

- N, M: size of the maze matrix
- NxM matrix: 0 path; 1 wall
- start and end points: sx, sy, ex, ey

➔ int maze[101][101], turn[101][101], dir[101][101];

➔ Do we need visited[][]?

➔ what should be initial value?

➔ Does starting point have a direction?

```
1000010
0010100
0000101
0110000
0000110
```

Minimum turns (P8121)

```
#define BIG 100000;
int maze[101][101], turn[101][101], dir[101][101];

int dx[4] = { 0, 0, 1, -1 };
int dy[4] = { 1, -1, 0, 0 };
```

```
    for(i = 0; i < n; i++)
        for(j = 0; j < m; j++) {
            cin >> maze[i][j];
            dir[i][j] = -1;
            turn[i][j] = BIG;
        }
```

```
            if (nx >= 0 && nx < n && ny >= 0 && ny < m && maze[nx][ny] == 0) { // can move
                if (turn[pt.x][pt.y] == 0 && dir[pt.x][pt.y] == -1) {  // start point
                    turn[nx][ny] = 0;
                    dir[nx][ny] = i;
                    q.push(PNT(nx, ny));
                    continue;
                }

```

```
                // not start point
                int nturn = (i == dir[pt.x][pt.y])? turn[pt.x][pt.y]: turn[pt.x][pt.y]+1;
                if (nturn < turn[nx][ny]) {
                    turn[nx][ny] = nturn;
                    dir[nx][ny] = i;
                    q.push(PNT(nx, ny));
                }
            }
        }
    }

    cout << turn[ex][ey] << endl;
```

Stack

XCamp

Scanf - faster alternative of cin

#include <cstdio>

int scanf (const char * format, ...);

Examples

```
printf ("Enter your full name: ");
scanf ("%s %s", firstName, lastName);
scanf ("%d", &n);// scanf requires the variable address
scanf ("%c",&ch); // scanf requires the variable address
```

Printf - faster alternative of cout

int printf (const char * format, ...);

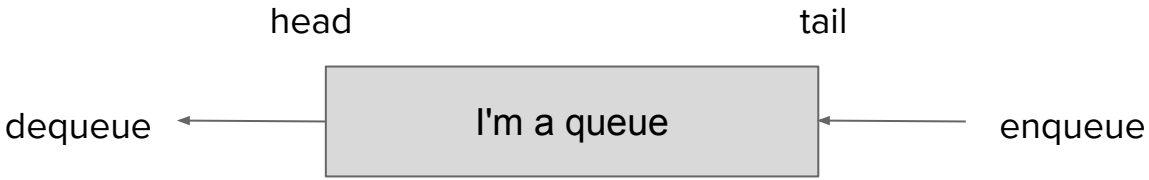
Examples

```
printf ("Characters: %c %c \n", 'a', 65); // ‘a’
printf ("Result: %d\n", n); // cout << "Result: " << n << endl;
printf ("Some different radices: %d %x %o %#x %#o \n",
100, 100, 100, 100, 100);
printf ("floats: %4.2f \n", 3.1416); //3.14
printf ("%s \n", "A string");
```

Practice: print out an array using printf

Quick Review: queue

- A basic data structure that simulating the literal real-life queue.
- First in, first out (FIFO)
- We use it in BFS



What's stack

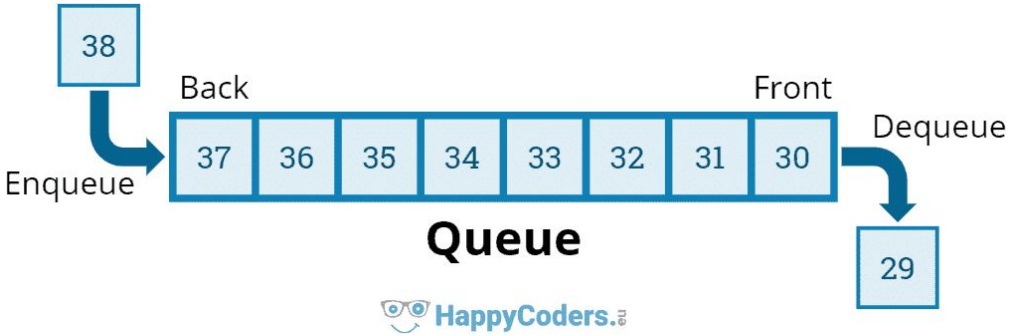
A linear data structure that stores elements. Two ends.

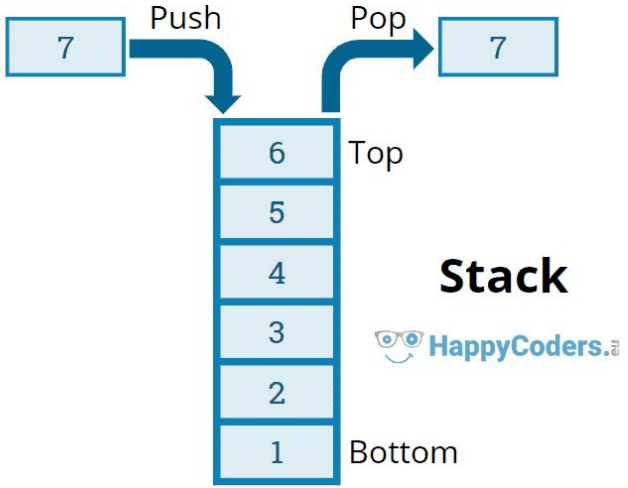
Push and pop are limited in one end only.

LIFO (Last In, First Out) while queue is FIFO

Why it matters? You will see later.

Stack examples: a book stack, browsing history,

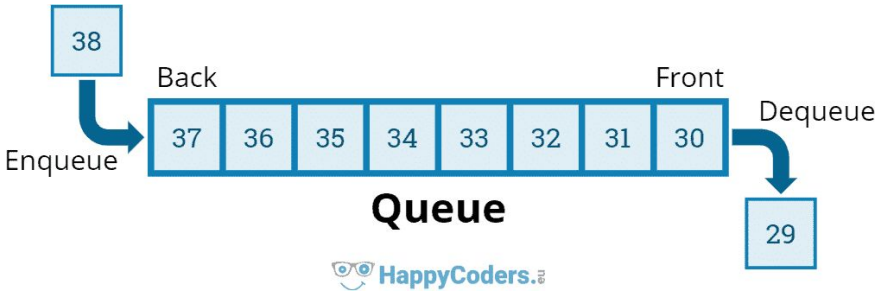




Important functions for stack

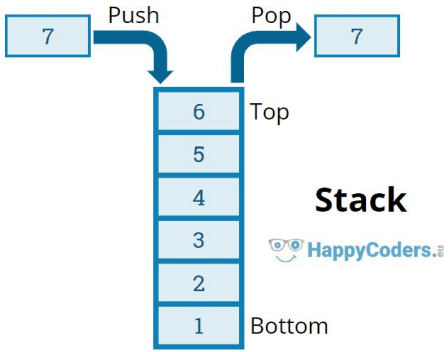
queue

- Empty
- Size
- Front
- Back
- Push
- Pop



stack

- Empty
- Size
- Push
- Top
- Pop

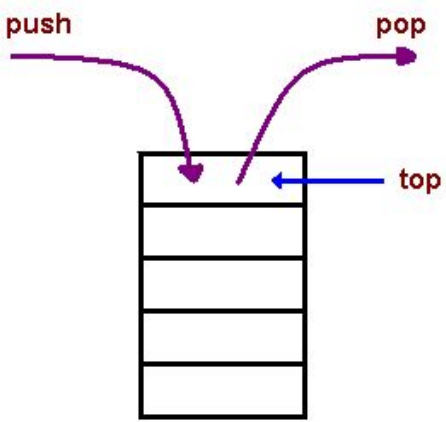


How to write codes to simulate stack?

- Push operation
- Pop operation
- CheckTop operation
- What basic data type can we use?

Simulate stack with array or vector

Using array or vector to simulate stack



Practice: Implement your stack (P7651)

Sample input // write your code to simulate stack using array or vector

```
5  //number of operations
+ 9  // + means push
- // - means pop
+ 10
+ -20
-
```

Sample output

```
10  // output elements in stack
```

- push()
- pop()
- top()
- empty()
- size()

```
vector<int> vec;
vec.insert(vec.begin(), m);
vec.erase(vec.begin());
```

Practice: https://xjoi.net/contest/4362/problem/3

Question 1

Use a stack that you implemented to reverse a list of numbers.

Input:
N
N numbers

Output
Reversed N numbers

Sample Input:
5
3 2 3 1 4

Sample Output:
4 1 3 2 3

std::stack<DataType>

```
#include<stack>

stack<int> s;

s.push(4);

cout << s.top();

s.pop();
```

https://www.cplusplus.com/reference/stack/stack/

Struct - constructor

```
struct point {
    int x, y;
    point(int a, int b) : x(a), y(b) {}
};
```

```
struct point {
    int x, y;
    point(int a, int b) {
        x = a;
        y = b;
    }
};
```

Constructor allows you to do

```
point p(10, 20);
```

```
stack<point> stk;
stk.push(point(x, y));
```

```
point p(10, 20);
```
Equals to:
```
point p;
p.x = 10;
p.y = 20;
```

```
stk.push(point(x, y));
```
Equals to:
```
point p;
p.x = 10;
p.y = 20;
stk.push(p);
```

Question 2 : Log analysis (P 7669)

// use std::stack

Input:
5 // 5 operations
0 3 // first number 0 means a push operation
0 4
1 // 1 means a pop operation
1
0 4
Output
4

Practice: https://xjoi.net/contest/4362/problem/4

Question 3

Compute the result of an expression, e.g., 4 + 3 - 2 + 3

All numbers are positive single digit integers or 0.

Operators are limited to +-

All expressions are expected to be legitimate.

Input is one line representing the expression.

Output is one line representing the result.

Question 4

Compute the result of an expression, e.g., 4 + 3 * 2 + 3

All numbers are positive single digit integers or 0.

Operators are limited to +-*/

All expressions are expected to be legitimate.

Input is one line representing the expression.

Output is one line representing the result.

Question 4

How can we take care of the priority of the operator?

5 + 4 * 3 - 2

How can our codes compute "4 * 3" before we compute "5 + "?

Question 4

How can we take care of the priority of the operator?

5 + 4 * 3 - 2

How can our codes compute "4 * 3" before we compute "5 + "?

Stack to help as a storage to help us "remember" and defer operations!

Question 4

Stack helps us defer the computation of an operator by storing the operator in one stack, and storing operands in another stack.

4 + 3 * 4 + 1

Operator Stack:

Operand Stack:

With hints above, please play with the two stacks above.

Stack Operations Details

4 + 3 * 4 + 1	Operator Stack(Left Bottom)	Operand Stack(Left Bottom)
4 + 3 * 4 + 1		4
4 + 3 * 4 + 1	+	4
4 + 3 * 4 + 1	+	4, 3
4 + 3 * 4 + 1	+, *	4, 3
4 + 3 * 4 + 1	+, *	4, 3, 4
4 + 3 * 4 + 1	+, *, +	4, 3, 4
4 + 3 * 4 + 1	+ (pop until no higher/equal operator present)	4, 12
4 + 3 * 4 + 1	+	16, 1
4 + 3 * 4 + 1		17

Question 5

Compute the result of an expression, e.g., 4 + 3 * (2 + 3)

Operands can be positive/negative, fractions/integers, or 0.

Operands can be multi-digit, such as 434.35.

Operators are limited to +-*/^

All expressions are expected to be legitimate.

1321 in XJOI.

How to handle the parenthesis?

If we see “(“ , we push it into the operator stack.

If we see “)” , we pop until we see the corresponding “(“.

HW Problem 4 (P1321) - Bonus Problem

2^3+5/3*(231-130)+10000

- Use two stacks for operator and operand
- How to read numbers (including decimal numbers)?
- When you see ‘)’, go back to finish calculation until you see ‘(‘ in stack
- Define priorities of operators (+, - , *, /, ^)
- Negative numbers
- Other corner cases?