    void push(T item) {

        list.add(item);

    }

    T pop() {

        // TODO: Remove an element on top of the stack

        T tmp = list.get(0);

        list.removeAt(0);

        return tmp;

    }

    T top() {

        // TODO: Get value of the element on top of the stack

        return list.get(0);

    }

    bool empty() {

        // TODO: Determine if the stack is empty

        return list.empty();

    }

    int size() {

        // TODO: Get the size of the stack

        return list.size();

    }

    void clear() {

        // TODO: Clear all elements of the stack

        list.clear();

    }

// iostream, vector and queue are included

// You can write helper methods

long long nthNiceNumber(int n) {

long long val;

queue<long long> qq;

qq.push(2);

qq.push(5);

while (n--){

val = qq.front();

qq.push(qq.front()\*10+2);

qq.push(qq.front()\*10+5);

qq.pop();

}

return val;

}

// iostream and queue are included

// Hint: Use a queue to simulate the process

int numberOfTheWinner(int N, int k) {

queue<int> qq;

for (int i = 1 ; i <= N ; i++) qq.push(i);

while(qq.size() != 1){

for (int i = 1 ; i <= k ; i++){

if (i!=k){

qq.push(qq.front());

qq.pop();

}

else qq.pop();

}

}

return qq.front();

}

int sumOfMaxSubarray(vector<int>& nums,int k) {

deque<int> dq;

long long sum = 0;

for (int i = 0 ; i < (int) nums.size(); i++){

while(!dq.empty() && nums[dq.back()] <= nums[i])

dq.pop\_back();

while(!dq.empty() && dq.front() <= i-k)

dq.pop\_front();

dq.push\_back(i);

if (i >= k-1) sum+=nums[dq.front()];

}

return sum;

}