CS3334 - Data Structures
Lab 1

Outline

- CS Online Judge – User Guidance
- Simple Exercises about Linked list and Stack
- Two OJ problems (755 and 740)

Access

- *http://acm.cs.cityu.edu.hk/oj2/*
- CSLab VPN is required
  - *automatic settled on all CS Lab computers*
  - *CSLab VPN connection: https://cslab.cs.cityu.edu.hk/services/cslab-vpn-sonicwall*
- User Account
  - *Need activation your account.* ***The account activation email*** *has sent to your registration email (from CityU CS OJ Administrator <cityuacm@outlook.com>)*
  - *contact TA Mr. Jiacheng HUANG if you have not received email*

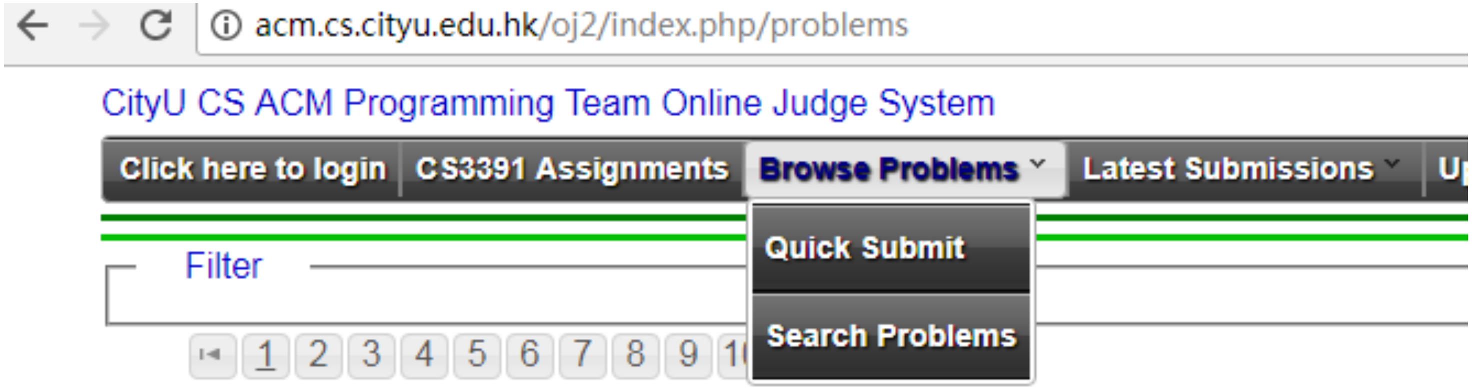
  (provide your student id and email address)

- Special Helper: Iusuf SHAKIRZIANOV

  (ishakirzi2-c@my.cityu.edu.hk)

Search Problems

- Browse Problems
  - or use problem id: http://acm.cs.cityu.edu.hk/oj2/index.php/p/755



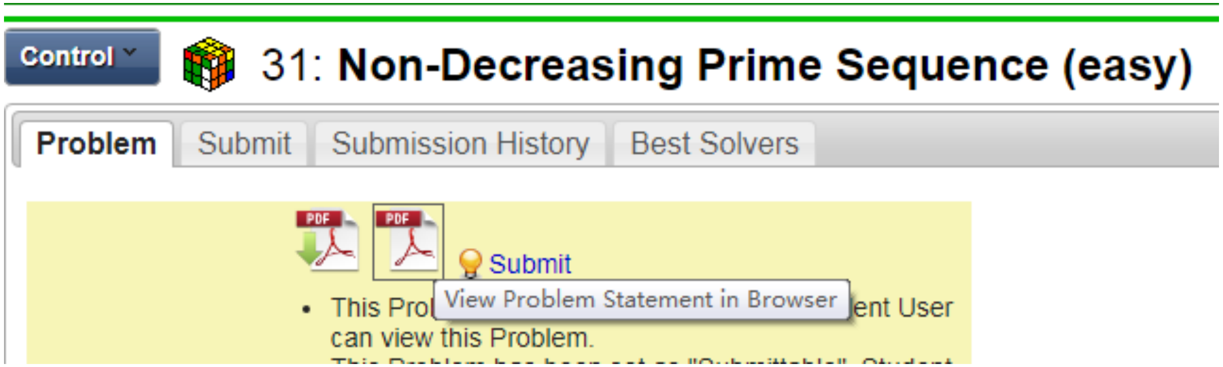
acm.cs.cityu.edu.hk/oj2/index.php/problems

CityU CS ACM Programming Team Online Judge System

- Search Problems
  - tag ‘CS3334_2024_Spring’

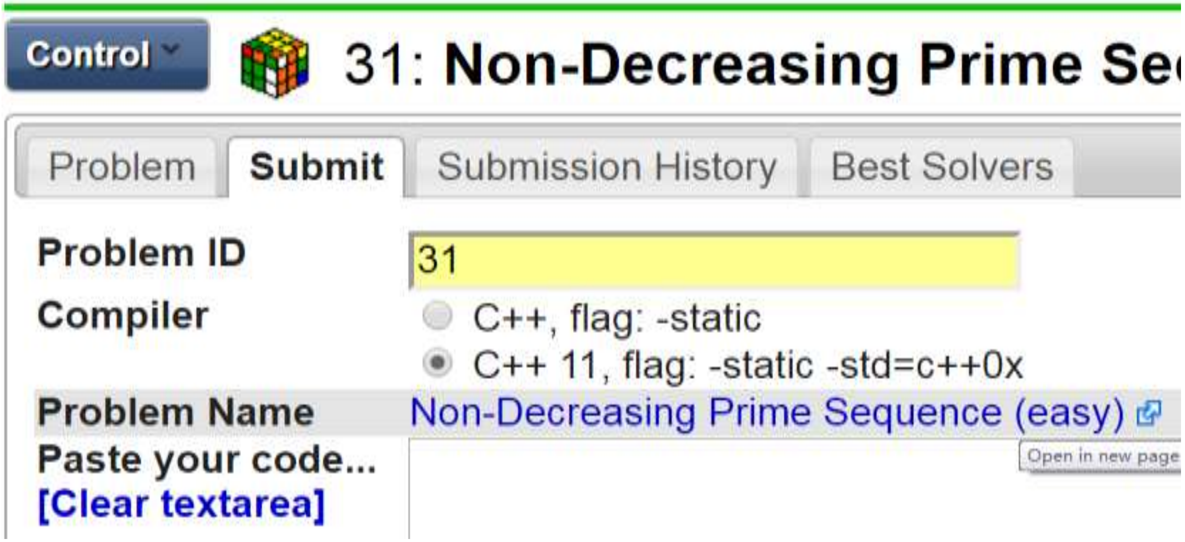
Submit Code

- Check Problem Statement


Control
31: Non-Decreasing Prime Sequence (easy)
Problem
Submit
Submission History
Best Solvers
Submit
View Problem Statement in Browser

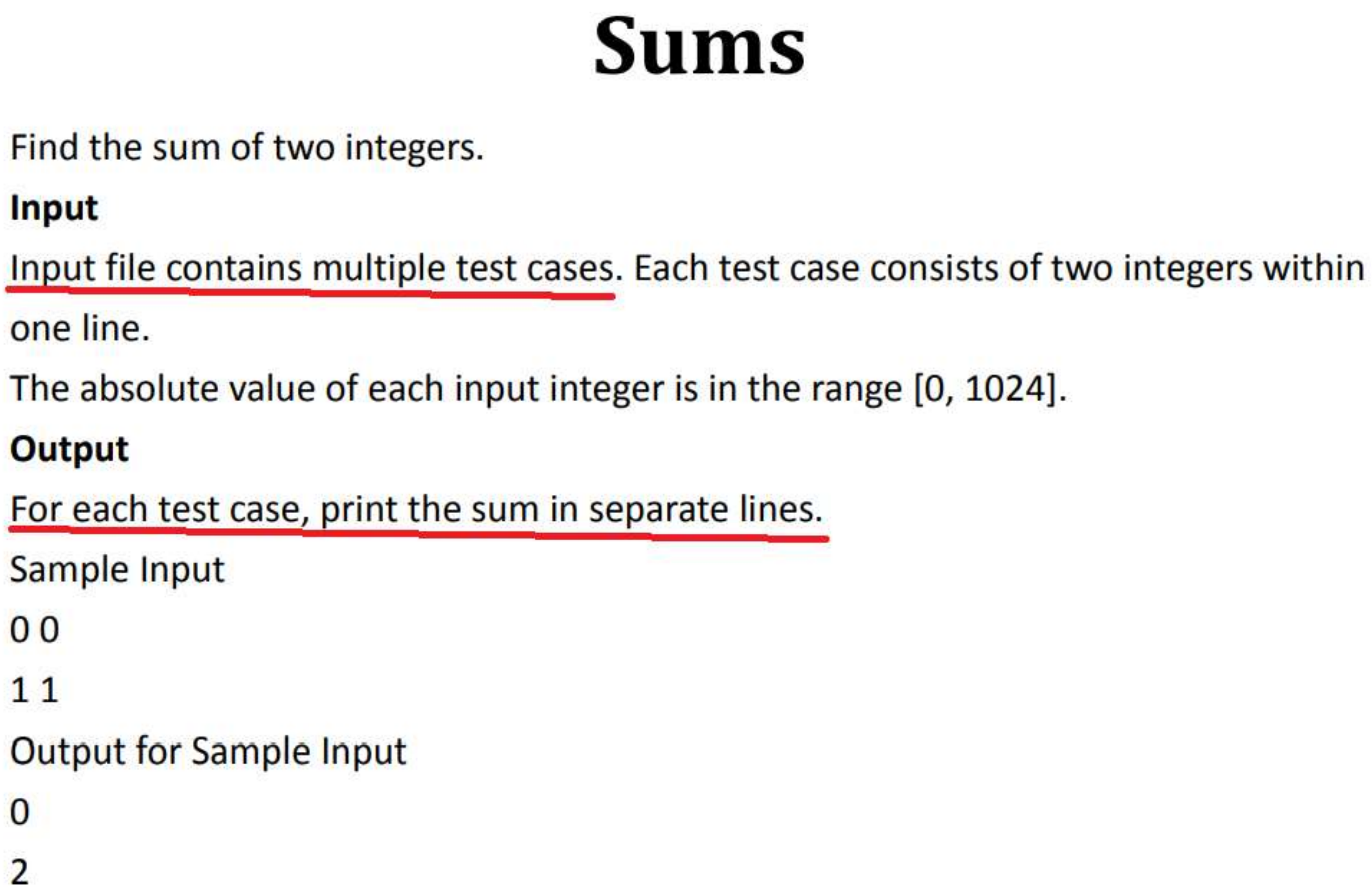
- This Pro ent User can view this Problem.

- Submit
  - compiler
  - paste code



Problem ID	31
Compiler	○ C++, flag: -static ● C++ 11, flag: -static -std=c++0x
Problem Name	Non-Decreasing Prime Sequence (easy)
Paste your code...	

Example problem OJ1



Sums

Find the sum of two integers.

Input

Input file contains multiple test cases. Each test case consists of two integers within one line.

The absolute value of each input integer is in the range [0, 1024].

Output

For each test case, print the sum in separate lines.

Sample Input

```
0 0
1 1
```

Output for Sample Input

```
0
2
```

Input/Output

```
//example
#include <iostream>
#include <sstream>
#include <string>
using namespace std;
int main(){
        string inputstr, outputstr;
        cin >> inputstr;
        outputstr = inputstr + " Hello";
        cout << outputstr << endl;
        return 0;
}
```

```
//read line-by-line
string linestr;
while ( getline(cin, linestr) )
```

```
//read from string
string input = “Hello World";
stringstream myStream(input);
myStream >> str1 >> str2;
```

Note: **space** and **line breaks** will be automatically ignored when reading.

Input/Output

- End of Input

```
while (cin >> str){
// output result
}
```

- Output Format (check problem statement)
  - for each test case, output the result in one line

```
while (cin >> caseinput1 >> caseinput2)
        cout << caseinput1 + caseinput2 << endl;
```

Verdict Information

- ❑ **In Queue (QU):** your submission code is in the queue, to be compiled.
- ❑ **Compile Error (CE):** The server could not compile your submission code. Of course, warning messages are not error messages.

  mostly it is caused by syntax errors:

  $ variable not defined,

  $ variable type not match,

  $ need include library
- ❑ **Accepted (AC):** OK! Your program is correct!
- ❑ **Presentation Error (PE):** Your program outputs are correct but are not presented in the correct way.

  Check for spaces line breaks, ‘ ’, ‘\n’, ‘comma’...

Verdict Information

- □ **Wrong Answer (WA):** output is not correct.
  $input format,
  $special case, any **tricky** test case?
- □ **Runtime Error (RE):** Your program failed during the execution.
  $ index overflow, a[-1], a[n+1].
  $ Stack/memory overflow, endless recursions, self call
- □ **Time Limit Exceeded (TLE):** Your program is not finished in required time.{time complexity, infinite loop, waiting for input,}
- □ **Memory Limit Exceeded (MLE):** Your program tried to use more memory than allowed.

static memory (array size) + memory for stack/queue during execution

- □ **Output Limit Exceeded (OLE):** Your program output too much than expected. {infinite loop}

Exercise 1 Manipulate List

Given a singly linked list, complete a function

insert(i, d) which inserts a new node with data *d* at position *i* of a Singly Linked List.

① 2 3 4

For example, ”7 5 3 1” with i=1, d=2 will become -> ”7 2 5 3 1 ")

2

If *i* is larger than the current list size, we do nothing.

position data

```
void List::insert(int i, int d)
{
            . . .
}
```

```
class ListNode
{
public:
      ListNode( int );
      ListNode( int, ListNode *);
      ListNode *get_Next()
      …
private:
      int data;
      ListNode *next;
};
```

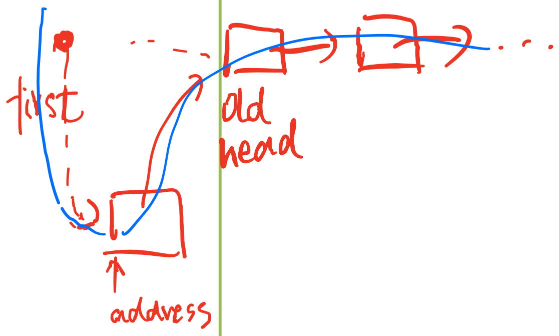
```
class List
{
public:
      List( String  );
      List();
      int size();
      ... //various member functions
private:
      ListNode *first;
      string name;
}
```

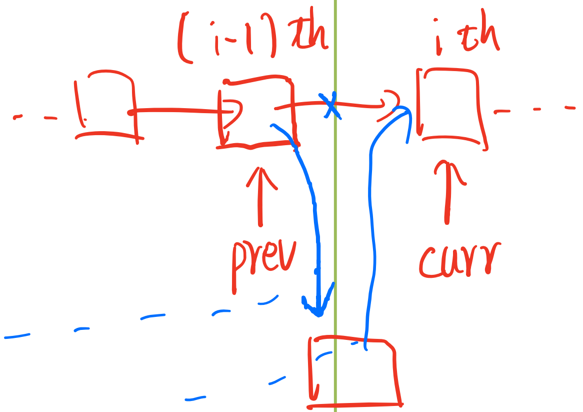
```
void List::insert(int i, int d)
{
    if (i < 0)
        return;

    ListNode* new_node = new ListNode(d);
    if (i == 0) {
        new_node->next = first;
        first = new_node;
        return;
    }

    ListNode* prev = nullptr; // singly list
    ListNode* curr = first;   // iterater
    int pos = 0;

    // Traverse to the (i-1)-th element
    while (curr != nullptr && pos < i) {
        prev = curr;
        curr = curr->get_Next();
        pos++;
    }
       pos == i

    if (curr != nullptr)
        prev->next = new_node;

    new_node->next = curr;
}
```

Check the parameter (not for grading)

→ Creat new node

New head node





Exercise 2 Manipulate List

Given a singly linked list, complete a function

reverse(i, j) which reverses elements from i-th element to j-th element(i,j inclusive).

For example, ”0 1 2 3 4 5” with i=1, j=3 will become -> ”0 3 2 1 4 5")

```
void List::reverse(int i, int j)
{
                . . .
}
```

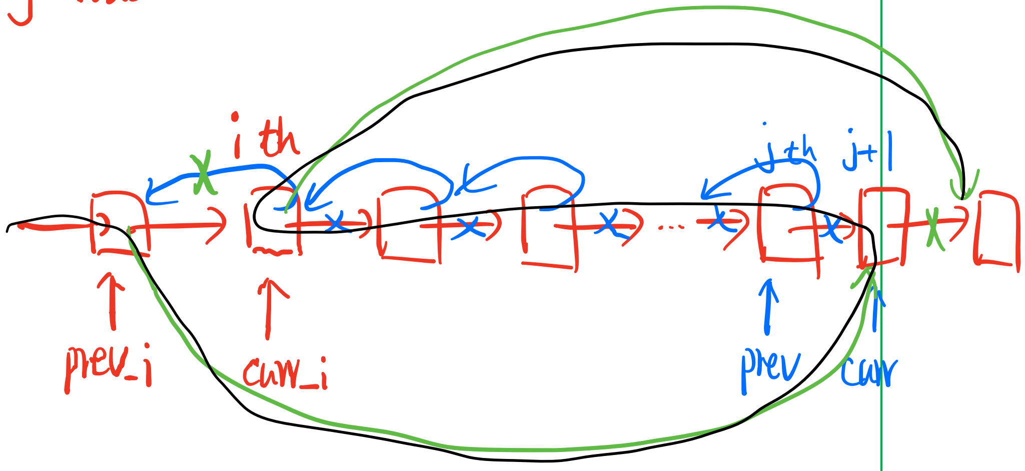
```
class ListNode
{
public:
      ListNode( int );
      ListNode( int, ListNode *);
      ListNode *get_Next()
      …
private:
      int data;
      ListNode *next;
};
```

```
class List
{
public:
     List( String  );
     List();
     int size();
     ... //various member functions
private:
     ListNode *first;
     string name;
}
```

```
void List::reverse(int i, int j)
{
   if (i >= j || first == nullptr || first->get_Next() == nullptr)
      return;

   ListNode* prev = nullptr;  ListNode* curr = first;
   int pos = 0;

   while (curr != nullptr && pos < i) {
      prev = curr;
      curr = curr->get_Next();
      pos++;
   }
   ListNode* prev_i = prev;  ListNode* curr_i = curr;

   // Reverse the elements from i-th to j-th
   ListNode* next_node = nullptr;
   while (curr != nullptr && pos <= j) {
      next_node = curr->get_Next();
      curr->next = prev;
      prev = curr;
      curr = next_node;
      pos++;
   }

   if (prev_i != nullptr)      prev_i->next = prev;
   else           first = prev;
   curr_i->next = curr;
}
```

singly list

record



Exercise 3 Stack

Given a stack, complete a function

delete(d) which removes all occurrences of an item *d* in a stack.

For example, ”~~5~~ 2 3 ~~5~~ 4” (4 is on top) with d = 5 will become -> ”2 3 4") (4 is on top)

top

top

```
// Stack.h
#include “stdlib.h”
{
        public class Stack
        {
                public:
                        Stack();
                        bool IsEmpty();
                        bool IsFull();
                        void push(int);
                        int pop();
                        int top();
                private:
                        …
                        // maybe array or linked based implementation
        };
}
```

```
void Stack::delete(int d)
{
            . . .
}
```

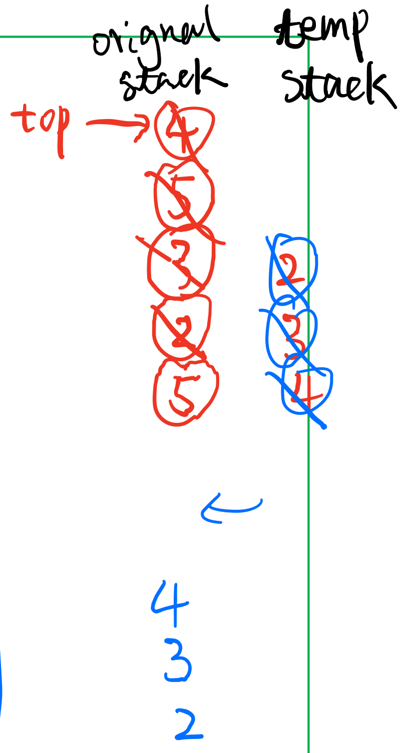
Exercise 3 Stack

Given a stack, complete a function

delete(d) which removes all occurrences of an item *d* in a stack.

For example, ”5 2 3 5 4” (4 is on top) with d = 5 will become -> ”2 3 4") (4 is on top)

```
void Stack::delete(int d)
{
    std::stack<int> tempStack;

    while (!IsEmpty())
    {
        int top = pop();
        if (top != d)
            tempStack.push(top);
    }

    while (!tempStack.empty())
    {
        int top = tempStack.top();
        tempStack.pop();
        push(top);
    }
}
```



740 Manipulate List

Description

Now there is a list consisting many integers. You are required to deal with the following operations on the list. operation

1. Insert : insert an integer at a certain position
2. Delete : delete the i-th element in the current list
3. Reverse : reverse elements from i-th element to j-th element(i,j inclusive).(eg:"1 2 3" -> "3 2 1")
4. Query : output the wanted element in the list

Input/Output

The test file consists of a **single** test case:

- The first line contains a number N (1<= N <= 1000) indicating the initial number of elements in the list.
- Then there will be N integers representing the initial elements in the list.
- Next line will be an integer Q(1 <= Q <= 50) which means the number of operations.
- After that, there will be Q lines of operations in the following format:

(1) 1 i val : insert "val" after the i-th element
(2) 2 i :delete element at i-th position
(3) 3 i j: reverse interval [i,j] of the list → Std :: list doesn't support
(4) 4 i: output the i-th element in the list

operator → operand

For each type (4) operation in the input, print the corresponding output in one line.

740 Manipulate List

Sample Input	Sample Output
6 N	4
1 2 3 4 5 6 initial elements in the list	5
12 Q	7
1 3 10	5
2 1	
4 4 Q operations	
3 3 5	
4 3	
2 2	
1 2 7	
3 1 4	
4 2	
2 5	
3 25	
4 4	

740 Manipulate List

Sample Input	Sample Output

6
1 2 3 4 5 6
12
1 3 10
2 1
4 4
3 3 5
4 3
2 2
1 2 7
3 1 4
4 2
2 5
3 25
4 4
	4
	5
	7
	5

1 2 3 4 5 6

10

1 2 10 3 4 5 7

2 10 3 4 5 7

2 10 3 4 5 7

2 10 5 4 3 7

755 Stock Market

stack | high | left | → always increasing

$O(N \times (1 + \text{length}(\text{stack}))) \sim O(N \times \text{length}(\text{stack})) < O(N^2)$

Description

Sherlock is a mad data scientist. Recently he drew a histogram using a dataset with size **N** from stock market showing the stock price on each day. Those days are consecutive. Can you help him find out the rectangle of the maximum area in the corresponding histogram?

stack

0. ϕ res = ~~0~~ ~~6~~ ~~8~~ 12

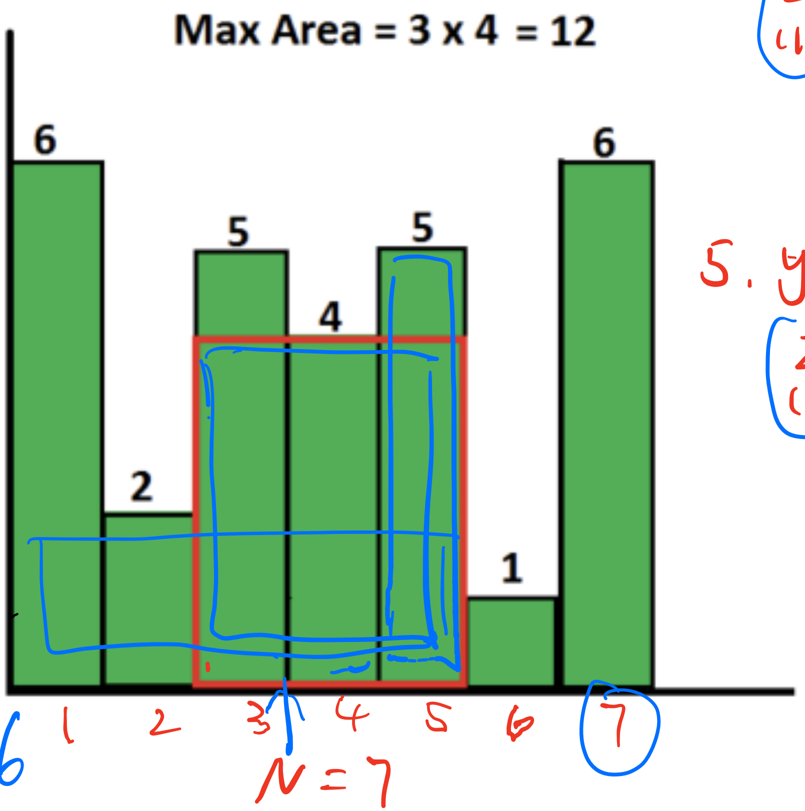
1. 6 (1) $1 \times 6 = 6$ res = 6

2. 6 (1) | 2 (2)

⇒ 2 (1) $2 \times 2 = 4$ res = max(4,6) = 6

3. 2 (1) | 5 (3) $5 \times (3-3+1) = 5$

$2 \times (3-1+1) = 6$



4. y = 4 x = 4

2 (1) ~~5 (3)~~ 4 (3)

$4 \times (4-3+1) = 8$

$2 \times (4-1+1) = 8$

5. y = 5 x = 5

2 (1) 4 (3) 5 (5)

$5 \times (5-5+1) = 5$

$4 \times (5-3+1) = 12$

$2 \times (5-1+1) = 10$

755 Stock Market

$\binom{7}{2}$ $\binom{N}{2}$

$O(N \cdot (n-1))$

$O(N^2)$

Input

The first line of input is an integer T (1 <= T <= 10) representing the number of test cases. Each test case will follow the format shown below:

The first line: An integer N showing the number of days in the dataset.

The second line: N integers p1, p2, ..., pN showing stock prices on each day.

(1 <= N<= 100000, 1 <= pi <= 100000).

Target : 10^9

10^5

$O(N^2) \sim (10^5)^2 \cdot 10 \sim 10^{11}$

T

very large

X

Output

For each case, print a single line containing the maximum area of the rectangle in the histogram.

monotonic stack

755 Stock Market

Sample Input	Sample Output

2
6
3 4 5 2 3 8
7
6 2 5 4 5 1 6
	12
	12

755 Stock Market

Sample Input	Sample Output

2
6
3 4 5 2 3 8
7
6 2 5 4 5 1 6
	12
	12

755 Stock Market

Sample Input	Sample Output

2
6
3 4 5 2 3 8
7
6 2 5 4 5 1 6
	12
	12

We will choose consecutive days with price {5 4 5} to get the max area 12 as in the figure.

Tips: the result may be bigger than the range of int, you may consider using long long int instead of int.