

# Fortinet - BackEnd

Q1.

## 1. Message Queues

In a modern distributed system, message queues are important components that provide communication between and coordination of the parts of the system. Which of the following are true?

Pick **ONE OR MORE** options

- ☐ Message queues make the system more decoupled.
- ☐ Message queues increase the reliability of the system.
- ☐ Message queues, in general, decrease the overall performance of the system.
- ☐ Message queues increase the complexity of the system architecture.

[Clear Selection](#)

- A,D

Q2.

## 2. Load Balancing

Load balancing is a technique used to distribute a load across multiple devices, servers for example. Often it is configured to distribute the load evenly. Which of the following are true?

Pick **ONE OR MORE** options

- ☐ Starving occurs when some resource, e.g. a server, does not process any requests, but others do.
- ☐ Any reasonable system has only one load balancer.
- ☐ A health check is a procedure that checks if the load is indeed balanced across resources.
- ☐ If a load balancer is not a bottleneck, then using it increases the throughput of the system.
- ☐ Sometimes it makes sense to have a load balancer that distributes the load in an uneven manner.

[Clear Selection](#)

- A, E

Q3.

### 3. Good URI Design

Which of the following are true regarding good *URI* design?

Pick **ONE OR MORE** options

☐ URIs should never be changed.

☐ URIs must be constructed by the client.

☐ URIs should be short in length.

☐ URIs should be case-sensitive.

☐ HTTP verbs should be used instead of operation names in URIs.

☐ Use spaces when designing a URI.

☐ Redirection must be used if a change in URI is required.

- A,C,E

Q4.

#### 4. Layout

---

If you want to wrap a block of text around an image, which CSS property would you use?

Pick **ONE** option

☐ wrap

☐ align

☐ push

☐ float

- B // not ok
- D // ok

Q5.

## 5. CSS Style

Choose the CSS style that most nearly creates the div element below:

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language.[1] Although most often used to set the visual style of web pages and user interfaces written in HTML and XHTML, the language can be applied to any XML document, including plain XML, SVG and XUL, and is applicable to rendering in speech, or on other media. Along with HTML and JavaScript, CSS is a cornerstone technology used by most websites to create visually engaging webpages, user interfaces for web applications, and user interfaces for many mobile applications.[2] CSS is designed primarily to enable the separation of document content from document presentation, including aspects such as the layout, colors, and fonts.[3] This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple HTML pages to share formatting by specifying the relevant CSS in a separate .css file, and reduce complexity and repetition in the structural content, such as semantically insignificant tables that were widely used to format pages before consistent CSS rendering was available in all major browsers. CSS makes it possible to separate presentation instructions from the HTML content in a separate file or style section of the HTML file. For each matching HTML element, it provides a list of formatting instructions. For example, a CSS rule might specify that "all heading 1 elements should be bold", leaving pure semantic HTML markup that asserts "this text is a level 1 heading" without formatting code such as a tag indicating how such text should be displayed.

<style>

<!--Choose the correct option to complete the code-->

</style>

<div></div>

<p>Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language.[1] Although most often used to set the visual style of web pages and user interfaces written in HTML and XHTML, the language can be applied to any XML document, including plain XML, SVG and XUL, and is applicable to rendering in speech, or on other media. Along with HTML and JavaScript, CSS is a cornerstone technology used by most websites to create visually engaging webpages, user interfaces for web applications, and user interfaces for many mobile applications.[2]

CSS is designed primarily to enable the separation of document content from document presentation, including aspects such as the layout, colors, and fonts.[3] This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple HTML pages to share formatting by specifying the relevant CSS in a separate .css file, and reduce complexity and repetition in the structural content, such as semantically insignificant tables that were widely used to format pages before consistent CSS rendering was available in all major browsers. CSS makes it possible to separate presentation instructions from the HTML content in a separate file or style section of the HTML file. For each matching HTML element, it provides a list of formatting instructions. For example, a CSS rule might specify that "all heading 1 elements should be bold", leaving pure semantic HTML markup that asserts "this text is a level 1 heading" without formatting code such as a <bold> tag indicating how such text should be displayed.</p>

Pick **ONE** option

Pick **ONE** option

☐

```
div {  
  shape-outside: ellipse(40%);  
  width: 400px;  
  height: 200px;  
  float: right;  
}
```

☐

```
div {  
  shape-outside: ellipse(40%);  
  width: 400px;  
  height: 200px;  
  float: left;  
}
```

☐

```
div {  
  shape-outside: ellipse(40%);  
  width: 400px;  
  height: 200px;  
  align:left;  
}
```

☐

```
div {  
  shape-outside: ellipse(40%);  
  width: 400px;  
  height: 200px;  
  align: right;  
}
```

☐

```
div {  
  shape-inside: ellipse(40%);  
  width: 400px;  
  height: 200px;  
  float: left;  
}
```



```
div {  
  shape-inside: ellipse(40%);  
  width: 400px;  
  height: 200px;  
  float: right;  
}
```

- B

Q6.

#### 6. Hidden Text Display

Initially, you have a text: "HTML, Javascript, CSS, Actionscript, CoffeeScript, VBScript, Silverlight are some of the front end languages". The text has a 5 pixel border that is displayed as given in the first figure, but the whole text is displayed only when the mouse hovers over it. Predict the code.

HTML, Javascript, CSS, Actionscript, CoffeeScript, VBScript

HTML, Javascript, CSS, Actionscript, CoffeeScript, VBScript, Silverlight are some of the front end languages

Pick **ONE** option

☐

```
p {  
  width: 500px;  
  border: 5px solid lightgreen;  
  overflow: hide;  
  color: black;  
}  
p: hover {  
  overflow: visible;  
}
```

☐

```
p {  
  white-space: nowrap;  
  width: 500px;  
  height: 500px;  
  border: 5px solid lightgreen;  
  overflow: hide;  
  color: black;  
}  
p: hover {  
  overflow: visible;  
}
```

☐

```
p { white-space: nowrap;  
  width: 500px;  
  border: 5px solid lightgreen;  
  overflow: hidden;  
  color: black;  
}  
p: hover {  
  overflow: visible;  
}
```

☐

```
p{
  white-space: nowrap;
  width: 500px;
  border: 5px lightgreen;
  overflow: hidden;
  color:black;
}
p:hover{
  overflow: visible;
}
```

☐

```
p.test2 {
  width: 500px;
  border: 5px solid lightgreen;
  color:black;
}
p:hover{
  overflow: visible;
}
```

- C



Q7.

## 7. Button Style

A button group is given below:



The color of the button must change when the mouse hovers over it as given below:



```
<button class="tabs">Domains</button>  
<button class="tabs">Contests</button>  
<button class="tabs">Progress</button>  
<button class="tabs">Leader Board</button>  
<button class="tabs">Jobs</button>
```

Figure out the style for the buttons.

☐

```
.tabs {  
  background-color: lightgreen;  
  color: white;  
  padding: 20px;  
  text-align: center;  
  font-size: 16px;  
  align: left;  
}  
.tabs:hover {  
  background-color: green;  
}
```

☐

```
.tabs {  
  background-color: lightgreen;  
  color: white;  
  border:none;  
  padding: 20px;  
  text-align: center;  
  font-size: 16px;  
  float: left;  
}  
.tabs:hover {  
  background-color: green;  
}
```

☐

```
.tabs {  
  background-color: lightgreen;  
  color: white;  
  border:none;  
  padding: 20px;  
  text-align: center;  
  font-size: 16px;  
  
}  
.tabs:hover {  
  background-color: green;  
}
```

Pick **ONE** option

☐


```
.tabs {  
  background-color: lightgreen;  
  color: white;  
  border:2px solid green;  
  padding: 20px;  
  text-align: center;  
  font-size: 16px;  
  float: left;  
}  
.tabs:hover {  
  background-color: green;  
}
```

☐

```
.tabs {  
  color: lightgreen;  
  color: white;  
  border:none;  
  padding: 20px;  
  text-align: center;  
  font-size: 16px;  
  float: left;  
}  
.tabs:hover {  
  background-color: green;  
}
```

☐

```
.tabs {  
  color: lightgreen;  
  color: white;  
  border:none;  
  padding: 20px;  
  text-align: center;  
  font-size: 16px;  
  align: left;  
}  
.tabs:hover {  
  background-color: green;  
}
```



```
.tabs {  
  background-color: lightgreen;  
  color: white;  
  padding: 20px;  
  text-align: center;  
  font-size: 16px;  
  float: left;  
}  
.tabs:hover {  
  background-color: green;  
}
```

- 我猜是3
- 实测：B

## 8. JavaScript: Verify Input Value

In this challenge, the task is to implement the function *makeInputVerifier* such that:

- it takes 2 integer arguments, *minimum* and *maximum*.
- returns a new function that we'll call *verify*.
- the function *verify* takes a single integer argument, *inputValue*, and does the following:
  - If *inputValue* is less than *minimum*, it returns 'Input is less than minimum value'.
  - If *inputValue* is greater than or equal to *minimum* and less than or equal to *maximum*, it returns 'Input is in range'.
  - If *inputValue* is greater than *maximum*, it returns 'Input is more than maximum value'.

For example, calling *makeInputVerifier*(3, 10) must return a function *verify*, such that calling *verify*(5) returns 'Input in range' because  $5 > 3$  (the minimum) and  $5 < 10$  (the maximum).

Your implementation of the function will be tested by a provided code stub on several input files. Each input file contains 3 integer parameters for the function calls. The *makeInputVerifier* function will be called with the first and second integer parameters (*minimum* and *maximum* respectively), and then the returned function will be called with the third parameter. The result of that latter call will be printed to the standard output by the provided code.

### Constraints

- $minimum \leq maximum$

Q8.

### ▼ Input Format For Custom Testing

The first line contains an integer, *minimum*, to be passed to the *makeInputVerifier* function.

The second line contains an integer, *maximum*, to be passed to the *makeInputVerifier* function.

The third line contains an integer, *inputValue*, to be passed to function returned by the *makeInputVerifier* function.

### ▼ Sample Case 0

#### Sample Input For Custom Testing

```
10
20
15
```

#### Sample Output

```
Input is in range
```

#### Explanation

Calling *makeInputVerifier*(10, 20) returns a function *verify*(15) that returns 'Input is in range' because  $15 > 10$  and  $15 < 20$ .

### ▼ Sample Case 1

#### Sample Input For Custom Testing

```
10
20
5
```

#### Sample Output

```
Input is less than minimum value
```

#### Explanation

Calling *makeInputVerifier*(10, 20) returns a function *verify*(5) that returns 'Input is less than minimum value' because  $5 < 10$ .

## ▼ Sample Case 2

### Sample Input For Custom Testing

```
10
20
25
```

### Sample Output

```
Input is more than maximum value
```

### Explanation

Calling `makeInputVerifier(10, 20)` returns a function `verify(25)` that returns 'Input is more than maximum value' because  $25 > 20$ .

```
1 ✓ 'use strict';
2
3 const fs = require('fs');
4
5
6 process.stdin.resume();
7 process.stdin.setEncoding('utf-8');
8
9 let inputString = '';
10 let currentLine = 0;
11
12 process.stdin.on('data', function(inputStdin) {
13     inputString += inputStdin;
14 });
15
16 process.stdin.on('end', function() {
17     inputString = inputString.split('\n');
18
19     main();
20 });
21
22 function readLine() {
23     return inputString[currentLine++];
24 }
25
26 function makeInputVerifier(minimum, maximum) {
27     // write your code here
28 }
29 ✓ function main() {
30     const ws = fs.createWriteStream(process.env.OUTPUT_PATH);
31
32     const min = parseInt(readLine().trim());
33     const max = parseInt(readLine().trim());
34     const verify = makeInputVerifier(min, max);
35
36     const input = parseInt(readLine().trim());
37     const result = verify(input);
38
39     ws.write(`${result}\n`);
40     ws.end();
```

```
const Flags = {
    SMALL: 'Input is less than minimum value',
    LARGE: 'Input is more than maximum value',
    OK: 'Input is in range',
}

function makeInputVerifier(minimum, maximum) {
    let result = Flags.OK
    return function (input) {
```

```

    if (input < minimum) {
      result = Flags.SMALL
    } else if (input > maximum) {
      result = Flags.LARGE
    }
    return result
  }
}

//test
const res = makeInputVerifier(5, 10)(7)
console.log(res)
console.log(makeInputVerifier(5, 10)(4))
console.log(makeInputVerifier(5, 10)(11))

```

Q9.

#### 9. Linux: Access Log Cleanup

Complete the file stub `/home/ubuntu/894065-linux-access-log-cleanup/script.sh` with one or more steps that do the following:

- Extract the archive `/home/ubuntu/894065-linux-access-log-cleanup/archive.tar.gz`.
- In all the extracted access log files, find all the records with an identified (not empty) IP address, then output the results to `/tmp/access.log` using the newline as the delimiter.

##### Note

- The completed solution will be evaluated in a new, clean environment. Be sure everything is in the `/home/ubuntu/894065-linux-access-log-cleanup` folder.
- All the tasks should be done within a simple `"sudo solve"` execution invoked from the question directory.
- You have `sudo` access.

##### Grading

- The execution result of `"sudo solve"` invoked from the question directory solves the task.

##### Server details

To complete the DevOps questions, you'll need to login to the server per details below:

```

Username: ubuntu
Server: 54.167.254.5
Password: gtmf0r
Command: ssh ubuntu@54.167.254.5

```

Continue

- `cd /home/ubuntu/894065-linux-access-log-cleanup`
- `tar -xf archive.tar.gz`
- `vim script.sh`
- `grep -r --exclude=access.log -E -o "([0-9]{1,3}\.){3}[0-9]{1,3}" /home/ubuntu/894065-linux-access-log-cleanup > access.log`



Q10.

10. Trip Query

A travel and tour company has two tables relating to customers: *FAMILIES* and *COUNTRIES*. Each tour offers a discount if a minimum number of people book at the same time.

Write a query to print the maximum number of discounted tours any one family in the *FAMILIES* table can choose from.

▼ Schema

There are 2 tables: *FAMILIES*, *COUNTRIES*.

FAMILIES		
Name	Type	Description
ID	STRING	Unique ID of the family.
NAME	STRING	Name of the primary contact.
FAMILY_SIZE	INTEGER	Size of the family.

COUNTRIES		
Name	Type	Description
ID	STRING	Unique ID of the country.
NAME	STRING	Name of the country.
MIN_SIZE	INTEGER	Minimum size group to get a discount.

### ▼ Sample Data Tables

FAMILIES		
ID	NAME	FAMILY_SIZE
c00dac11bde74750b4d207b9c182a85f	Alex Thomas	9
eb6f2d3426694667ae3e79d6274114a4	Chris Gray	2

COUNTRIES		
ID	NAME	MIN_SIZE
023fd23615bd4ff4b2ae0a13ed7efec9	Bolivia	2
be247f73de0f4b2d810367cb26941fb9	Cook Islands	4
3e85ab80a6f84ef3b9068b21dbcc54b3	Brazil	4

### Sample Output

3

### Explanation

The Thomas family can choose from any of the 3 tours and qualify for the discount. The Gray family only qualifies for 1.

- `select COUNT(*) from COUNTRIES where MIN_SIZE < (select FAMILY_SIZE FROM FAMILIES order by FAMILY_SIZE desc limit 1)`
-

Q11.

### 11. List Customer and Product Without Sale

Using the UNION operator, in one list return all customers who do not have an invoice and all products that were not sold.

For each customer without an invoice, return:

- the string *customer*
- the customer *id*
- the *customer\_name*

Solve question 3

For each product without an invoice, return:

- the string *product*
- the *product id*
- the *product\_name*

The columns must be in the order shown, but row order does not matter.

Table definitions and a data sample are given below.

#### ▼ Schema

Table: customer

column name	column type	key / NULL
id	int	PK
customer_name	varchar(255)	
city_id	int	FK
customer_address	varchar(255)	
contact_person	varchar(255)	N
email	varchar(128)	
phone	varchar(128)	

Table: product

column name	column type	key / NULL
id	int	PK
sku	varchar(32)	
product_name	varchar(128)	
product_description	text	
current_price	decimal(8,2)	
quantity_in_stock	int	

**Table: invoice**

column name	column type	key / NULL
id	int	PK
invoice_number	varchar(255)	
customer_id	int	FK
user_account_id	int	
total_price	decimal(8,2)	
time_issued	varchar	N
time_due	varchar	N
time_paid	varchar	N
time_canceled	varchar	N
time_refunded	varchar	N

*invoice.customer\_id references customer.id*

**Table: invoice\_item**

column name	column type	key / NULL
id	int	PK
invoice_id	int	FK
product_id	int	FK
quantity	int	
price	decimal(8,2)	
line_total_price	decimal(8,2)	

*invoice\_item.invoice\_id references invoice.id*

*invoice\_item.product\_id references product.id*

▼ Sample Data Tables

Table: customer

id	customer_name	city_id	customer_address	contact_person	email	phone
1	Drogerie Wien	1	Deckergasse 15A	Emil Steinbach	emil@drogeriewien.com	094234234
2	Cosmetics Store	4	Watling Street 347	Jeremy Corbyn	jeremy@c-store.org	093923923
3	Kosmetikstudio	3	Rothenbaumchaussee 53	Willy Brandt	willy@kosmetikstudio.com	0941562222
4	Neue Kosmetik	1	Karlsplatz 2	NULL	info@neuekosmetik.com	094109253
5	Bio Kosmetik	2	Motzstraße 23	Clara Zetkin	clara@biokosmetik.org	093825825
6	K-Wien	1	Kärntner Straße 204	Maria Rauch-Kallat	maria@kwien.org	093427002
7	Natural Cosmetics	4	Clerkenwell Road 14B	Glenda Jackson	glena.j@natural-cosmetics.com	093555123
8	Kosmetik Plus	2	Unter den Linden 1	Angela Merkel	angela@k-plus.com	094727727
9	New Line Cosmetics	4	Devonshire Street 92	Oliver Cromwell	oliver@nlc.org	093202404

Table: product

id	sku	product_name	product_description	current_price	quantity_in_stock
1	330120	Game Of Thrones - URBAN DECAY	Game Of Thrones Eyeshadow Palette	65	122
2	330121	Advanced Night Repair - ESTEE LAUDER	Advanced Night Repair Synchronized Recovery Complex II	98	51
3	330122	Rose Deep Hydration - FRESH	Rose Deep Hydration Facial Toner	45	34
4	330123	Pore-Perfecting Moisturizer - TATCHA	Pore-Perfecting Moisturizer & Cleanser Duo	25	393
5	330124	Capture Youth - DIOR	Capture Youth Serum Collection	95	74
6	330125	Slice of Glow - GLOW RECIPE	Slice of Glow Set	45	40
7	330126	Healthy Skin - KIEHL S SINCE 1851	Healthy Skin Squad	68	154
8	330127	Power Pair! - IT COSMETICS	IT is Your Skincare Power Pair! Best-Selling Moisturizer & Eye Cream Duo	80	0
9	330128	Dewy Skin Mist - TATCHA	Limited Edition Dewy Skin Mist Mini	20	281
10	330129	Silk Pillowcase - SLIP	Silk Pillowcase Duo + Scrunchies Kit	170	0

Table: invoice

id	invoice_number	customer_id	user_account_id	total_price	time_issued	time_due
1	in_25181b07ba800c8d2fc967fe991807d9	7	4	1436	7/20/2019 3:05:07 PM	7/27/2019 3:05:07 PM
2	8fba000fd456b27502b9f81e9d52481	9	2	1000	7/20/2019 3:07:11 PM	7/27/2019 3:07:11 PM
3	3b6638118246b6bcfd3dfcd9be487599	3	2	360	7/20/2019 3:06:15 PM	7/27/2019 3:06:15 PM
4	dfe7f0a01a682196cac0120a9adbb550	5	2	1675	7/20/2019 3:06:34 PM	7/27/2019 3:06:34 PM
5	2a24cc2ad4440d698878a0a1a71f70fa	6	2	9500	7/20/2019 3:06:42 PM	7/27/2019 3:06:42 PM
6	cbd304872ca6257716bcab8fc43204d7	4	2	150	7/20/2019 3:08:15 PM	7/27/2019 3:08:15 PM

time_due	time_paid	time_canceled	time_refunded
7/27/2019 3:05:07 PM	7/25/2019 9:24:12 AM	NULL	NULL
7/27/2019 3:07:11 PM	7/20/2019 3:10:32 PM	NULL	NULL
7/27/2019 3:06:15 PM	7/31/2019 9:22:11 PM	NULL	NULL
7/27/2019 3:06:34 PM	NULL	NULL	NULL
7/27/2019 3:06:42 PM	NULL	7/22/2019 11:17:02 AM	NULL
7/27/2019 3:08:15 PM	7/27/2019 1:42:45 PM	NULL	7/27/2019 2:11:20 PM

Table: invoice\_item

id	invoice_id	product_id	quantity	price	line_total_price
1	1	1	20	65	1300
2	1	7	2	68	136
3	1	5	10	100	1000
4	3	10	2	180	360
5	4	1	5	65	325
6	4	2	10	95	950
7	4	5	4	100	400
8	5	10	100	95	9500
9	6	4	6	25	150

The first 2 lines of the result should be:

category ("customer" or "product") id (customer.id or product.id) name (customer\_name or product\_name)

-> customer 2 Cosmetics Store

-> product 9 Dewy Skin Mist - TATCHA

- `SELECT 'customer' as category,id,customer_name FROM customer`
- `WHERE id NOT IN(SELECT customer_id FROM invoice)`
- `UNION`
- `SELECT 'product' as category,id,product_name FROM product`
- `WHERE id NOT IN(SELECT product_id FROM invoice_item);`
- 

//Test passed:

```
SELECT 'customer' as category,id,customer_name FROM customer
WHERE id NOT IN(SELECT customer_id FROM invoice)
UNION
SELECT 'product' as category,id,product_name FROM product
WHERE id NOT IN(SELECT product_id FROM invoice_item);
```



## Q12.

### 12. Simple Max Difference

In securities research, an analyst will look at a number of attributes for a stock. One analyst would like to keep a record of the highest positive spread between a closing price and the closing price on any prior day in history. Determine the maximum positive spread for a stock given its price history. If the stock remains flat or declines for the full period, return -1.

#### Example 0

$px = [7, 1, 2, 5]$

Calculate the positive difference between each price and its predecessors:

- At the first quote, there is no earlier quote to compare to.
- At the second quote, there was no earlier price that was lower.
- At the third quote, the price is higher than the second quote:
  - $2 - 1 = 1$
- For the fourth quote, the price is higher than the third and the second quotes:
  - $5 - 2 = 3$
  - $5 - 1 = 4$ .
- The maximum difference is 4.

#### Example 1

$px = [7, 5, 3, 1]$

- The price declines each quote, so there is never a difference greater than 0. In this case, return -1.

#### Function Description

Complete the function *maxDifference* in the editor below.

*maxDifference* has the following parameters:

*int px[n]*: an array of stock prices (quotes)

Returns:

*int*: the maximum difference between two prices as described above

#### Constraints

- $1 \leq n \leq 10^5$
- $-10^5 \leq px[i] \leq 10^5$

### ▼ Input Format For Custom Testing

Locked stub code reads input from stdin and passes it to the function.

The first line contains an integer,  $n$ , denoting the number of elements in the array  $px$ .  
Each of the next  $n$  lines contains an integer,  $px[i]$ .

### ▼ Sample Case 0

#### Sample Input For Custom Testing

STDIN	Function
7	$px[]$ size $n = 7$
2	$px = [2, 3, 10, 2, 4, 8, 1]$
3	
10	
2	
4	
8	
1	

#### Sample Output

8

#### Explanation

Calculate the positive difference between each price quote and the previous ones :

- There is no predecessor for the first quote.
- At the second quote, the price is higher than the first quote:
  - $px[1] - px[0] = 3 - 2 = 1$
- At the third quote, the price is higher than the first and second quotes:
  - $px[2] - px[1] = 10 - 3 = 7$
  - $px[2] - px[0] = 10 - 2 = 8$
- At the fifth quote, the price is higher than the first and second quotes:
  - $px[4] - px[1] = 4 - 3 = 1$
  - $px[4] - px[0] = 4 - 2 = 2$
- At the sixth quote, the price is higher than the first, second, fourth and fifth quotes:
  - $px[5] - px[0] = 8 - 2 = 6$
  - $px[5] - px[1] = 8 - 3 = 5$
  - $px[5] - px[3] = 8 - 2 = 6$
  - $px[5] - px[4] = 8 - 4 = 4$
- The maximum difference is 8.

### ▼ Sample Case 1

#### Sample Input For Custom Testing

STDIN	Function
6	→ px[] size n = 6
7	→ px = [7, 9, 5, 6, 3, 2]
9	
5	
6	
3	
2	

#### Sample Output

2

#### Explanation

Calculate the positive difference between each quote and the previous ones :

- The second quote, the price is higher than the first:
  - $px[1] - px[0] = 9 - 7 = 2$
- After that, the prices decline steadily.
- The maximum difference is 2.

```
1 > import java.io.*; ...
14
15 class Result {
16
17     /*
18      * Complete the 'maxDifference' function below.
19      *
20      * The function is expected to return an INTEGER.
21      * The function accepts INTEGER_ARRAY px as parameter.
22      */
23
24     public static int maxDifference(List<Integer> px) {
25         // Write your code here
26
27     }
28
29 }
30
31 > public class Solution { ...
```

```
1  class Solution {  
2  public:  
3      int maxProfit(vector<int>& prices) {  
4          int n = prices.size();  
5          int dp_i_0 = 0, dp_i_1 = INT_MIN;  
6  
7          for (int i = 0; i < n; i++) {  
8              dp_i_0 = max(dp_i_0, dp_i_1 + prices[i]);  
9              dp_i_1 = max(dp_i_1, -prices[i]);  
10         }  
11  
12         return dp_i_0;  
13     }  
14 };  
•  
•
```

## Q13.

### 13. Crashing stones

Each day a quarry-worker is given a pile of stones and told to reduce the larger stones into smaller ones. The worker must smash the stones together to reduce them, and is told to always pick up the largest two stones and smash them together. If the stones are of equal weight, they both disintegrate entirely. If one is larger, the smaller one is disintegrated and the larger one is reduced by the weight of the smaller one. Eventually, there is either one stone left that cannot be broken, or all of the stones have been smashed. Determine the weight of the last stone, or return 0 if there is none.

#### Example

*weights* = [1,2,3,6,7,7].

The worker always starts with the two largest stones. In this case, the two largest stones have equal weights of 7 so they both disintegrate when smashed. Next the worker smashes weights 3 and 6. The smaller one is destroyed and the larger weighs  $6 - 3 = 3$  units. Then, weights 3 and 2 are smashed together, which leaves a stone of weight 1. This is smashed with the last remaining stone of weight 1. There are no stones left, so the remaining stone weight is 0.

#### Function Description

Complete the function *lastStoneWeight* in the editor below. The function must return an integer that denotes the weight of the last stone, or 0 if all stones shattered into dust.

*lastStoneWeight* has the following parameter(s):

*int weights[n]*: an array of integers indicating the weights of each stone

#### Constraints

- $1 \leq n \leq 10^5$
- $1 \leq weights[i] \leq 10^9$

#### ▼ Input Format for Custom Testing

The first line contains an integer *n* indicating the size of the array *weights*.  
Each of the next *n* lines contains an integer *weights[i]*.

#### ▼ Sample Case 0

##### Sample Input

STDIN	Function
-----	-----
3	→ weights[] size n = 3
2	→ weights = [2, 4, 5]
4	
5	

##### Sample Output

1

##### Explanation

First the worker takes stones with weights 4 and 5 and crashes them into each other. The first one disintegrates completely. The second one is reduced to a weight of 1. Next the worker crashes that stone into the last stone with weight 2. The smaller stone disintegrates, and the last stone is reduced to a weight of 1.

```

class Solution {
    public int lastStoneWeight(int[] stones) {
        PriorityQueue<Integer> pq = new PriorityQueue<Integer>((a, b) -> b - a);
        for (int stone : stones) {
            pq.offer(stone);
        }

        while (pq.size() > 1) {
            int a = pq.poll();
            int b = pq.poll();
            if (a > b) {
                pq.offer(a - b);
            }
        }
        return pq.isEmpty() ? 0 : pq.poll();
    }
}

```

```

class Result {
    /*
     * Complete the 'lastStoneWeight' function below.
     *
     * The function is expected to return an INTEGER.
     * The function accepts INTEGER_ARRAY weights as parameter.
     */

    public static int lastStoneWeight(List<Integer> weights) {
        // Write your code here
        PriorityQueue<Integer> pq = new PriorityQueue<Integer>((a, b) -> b - a);

        for(Integer weight : weights){
            pq.offer(weight);
        }

        while(pq.size() > 1){
            int a = pq.poll();
            int b = pq.poll();

            if(a > b){
                pq.offer(a - b);
            }
        }
        return pq.isEmpty() ? 0 : pq.poll();
    }
}

```

# Binance

Q1.

- 

Q2.

- 

Q3.

- 

Q4.

- 

Q5.

- 

Q6.

- 

Q7.

-

