Email:	arif.khalid@u.nus.edu
Test Name:	**[TikTok] SG Intern Assessment 2023 Start - 4 Dec to 8 Dec (Front-end) - Practice Session**
Taken On:	6 Dec 2023 09:29:45 IST
Time Taken:	77 min 32 sec/ 42000 min
Invited by:	Prepkit
Skills Score:	JavaScript (Intermediate) 75/75 Problem Solving (Intermediate) 150/150
Tags Score:	Adjacency Matrix 5/5 Algorithms 75/75 Arrays 75/75 Binary Search 75/75 Coding 75/75 Data Structures 80/80 Dijkstra's Algorithm 5/5 Easy 5/5 Graphs 5/5 Hard 5/5 Interviewer Guidelines 75/75 Javascript 80/80 Medium 225/225 Problem Solving 75/75 Programming Fundamentals 5/5 Prototypal inheritance 75/75 Strings 75/75 Theme: Finance 75/75

100%

235/235

scored in **[TikTok] SG Intern Assessment 2023 Start - 4 Dec to 8 Dec (Front-end) - Practice Session** in 77 min 32 sec on 6 Dec 2023 09:29:45 IST

Recruiter/Team Comments:

No Comments.

Plagiarism flagged

We have marked questions with suspected plagiarism below. Please review it in detail here -

	Question Description	Time Taken	Score	Status
Q1	Product Recommendations > **Multiple Choice**	7 min 4 sec	5/ 5	✓

Q2	**Dequeue operations** › **Multiple Choice**	1 min 24 sec	5/ 5	✓
Q3	**User-Friendly Password System** › **Coding**	28 min 49 sec	75/ 75	✓
Q4	**Profit Targets** › **Coding**	12 min 33 sec	75/ 75	✓
Q5	**JavaScript: Activity List** › **Coding**	27 min 35 sec	75/ 75	!

QUESTION 1

✓

Correct Answer

Score 5

Product Recommendations › Multiple Choice

Hard | Programming Fundamentals | Javascript | Dijkstra's Algorithm | Graphs | Adjacency Matrix

QUESTION DESCRIPTION

An e-commerce company has a platform that uses an algorithm to suggest products to users. It uses a graph-based approach, where each node represents a product, and the edges between them represent similarities. To avoid recommending too many similar items, they implement a modified version of Dijkstra's algorithm that takes into account the similarity score.

Here is the JavaScript code snippet:

```
function dijkstra(graph, startProduct) {
    let n = graph.length;
    let minDistances = new Array(n).fill(Infinity);
    let visited = new Array(n).fill(false);

    minDistances[startProduct] = 0;

    for (let i = 0; i < n; i++) {
        let minIndex = -1;
        for (let j = 0; j < n; j++) {
            if (!visited[j] && (minIndex === -1 || minDistances[j] <
minDistances[minIndex])) {
                minIndex = j;
            }
        }
        if (minDistances[minIndex] === Infinity) {
            break;
        }
        visited[minIndex] = true;

        for (let j = 0; j < n; j++) {
            if (graph[minIndex][j] !== 0) {
                let potentialDist = minDistances[minIndex] +
graph[minIndex][j];
                if (potentialDist < minDistances[j]) {
                    minDistances[j] = potentialDist;
                }
            }
        }
    }
    return minDistances;
}
```

Given an adjacency matrix that represents the similarity scores between different products:

```
let graph = [
    [0, 2, 0, 1, 0],
    [2, 0, 3, 0, 0],
    [0, 3, 0, 4, 0],
    [1, 0, 4, 0, 5],
    [0, 0, 0, 5, 0]
];
```

What is the output when the recommendation system calls dijkstra*(graph, 0);* to find the least similar products to the current product *(Product 0)*?

INTERVIEWER GUIDELINES

The adjacency matrix 'graph' represents the similarity scores between different products. Each row and column index corresponds to a product, and the value at graph[i][j] represents the similarity score between product 'i' and product 'j'. A higher score indicates higher similarity. The Dijkstra's algorithm in this case works by finding the path of least similarity (i.e., the path with the lowest total score).
So, starting from product 0, the algorithm calculates the least total similarity score to reach every other product. Hence, the output [0, 2, 5, 1, 6] represents the least total similarity scores from product 0 to every other product.

CANDIDATE ANSWER

Options: (Expected answer indicated with a tick)

✓ ● [0, 2, 5, 1, 6]

○ [0, 1, 4, 2, 7]

○ [0, 2, 3, 1, 5]

○ [0, 2, 3, 1, 6]

No Comments

QUESTION 2

✓ Correct Answer

Score 5

Dequeue operations › Multiple Choice

Easy | Data Structures

QUESTION DESCRIPTION

What will be the output when these operations are performed on a Doubly Ended Queue?

1. Insertfront(1);
2. Insertfront(2);
3. Insertrear(3);
4. Insertrear(4);
5. Deletefront();
6. Insertfront(5);
7. Deleterear();
8. Display();

INTERVIEWER GUIDELINES

Here is what dequeue will look like after each operation:
1
2 1
2 1 3
2 1 3 4
1 3 4
5 1 3 4
5 1 3

CANDIDATE ANSWER

Options: (Expected answer indicated with a tick)

○ 1, 2, 3

✓ ● 5, 1, 3

○ 4, 2, 3

○ None of the above

No Comments

QUESTION 3

✓

Correct Answer

Score 75

User-Friendly Password System › Coding Medium Strings

QUESTION DESCRIPTION

A website is programming an authentication system that will accept a password either if it's the correct password *or* if it's the correct password with a single character appended to it. In this challenge, your task is to implement such a system, specifically using a hashing function. Given a list of events in which either a password is set or authorization is attempted, determine if each authorization attempt will be successful or not.

The hashing function that will be used in this problem is as follows. Let f(x) be a function that takes a character and returns its decimal character code in the ASCII table. For instance f('a') = 97, f('B') = 66, and f('9') = 57. (You can find all ASCII character codes here: ASCII table.) Then, let h(s) be the hashing function that takes a string and hashes it in the following way, where $p = 131$ and $M = 10^9+7$:

$h(s) := (s[0]*P^{(n-1)} + s[1]*P^{(n-2)} + s[2]*P^{(n-3)} + ... + s[n-2]*P + s[n-1]) \bmod M$

For instance, if *s = "cAr1"*, then the formula would be as follows:

$h(s) = (f('c')*131^3 + f('A')*131^2 + f('r')*131 + f('1')) \bmod 10^9+7 = 223691457$

Your system will be tested on *q* event types, each of which will be one of the following:

1. setPassword(*s*) := sets the password to *s*
2. authorize(*x*) := tries to sign in with integer *x*. This event must return 1 if *x* is either the hash of the current password *or* the hash of the current password with a single character appended to it. Otherwise, this event must return 0.

Consider the following example. There are 6 events to be handled:

1. setPassword("cAr1")
2. authorize(223691457)
3. authorize(303580761)
4. authorize(100)
5. setPassword("d")
6. authorize(100)

As we know from the above example, h("cAr1") = 223691457, so the second event will return 1. The third event will also return 1 because 303580761 is the hash value of the string "cAr1a", which is equal to the current password with the character 'a' appended to it. The fourth event will return 0 because 100 is not a hash of the current password or of the current password with a single character appended to it. In the fifth event, the current password is set to "d", and the sixth event will return 1 because h("d") = 100. Therefore, the array you would return is [1, 1 0, 1], corresponding to the success or failure of the authorization events.

Function Description

Complete the function *authEvents* in the editor below.

authEvents has the following parameter(s):

string *events[q][2]*: a 2-dimensional array of strings denoting the event types and event parameters

Returns:

int[number of authorize events]: an array of integers, either 1 or 0, corresponding to the success (1) or failure (0) of each authorization attempt

Constraints

- $2 \le q \le 10^5$
- 1 ≤ length of $s \le 9$, where s is a parameter of the setPassword event
- $0 \le x < 10^9+7$, where x is the integer value of the parameter of the authorize event
- The first event will always be a setPassword event.
- There will be at least one authorize event.
- *s* contains only lowercase and uppercase English letters and digits.

▼ Input Format Format for Custom Testing

In the first line, there is a single integer, *q*, denoting the number of rows in *events*.

In the second line, there is a single integer, 2, denoting the number of columns in *events*.

Each line *i* of the *q* subsequent lines (where 0 ≤ *i* < *q*) contains two space-separated strings—*events[i][0]* denoting the event type ("setPassword" or "authorize") and *events[i][1]* denoting the event parameter (*s* or *x*.)

▼ Sample Case 0

Sample Input

```
4
2
setPassword 000A
authorize 108738450
authorize 108738449
authorize 244736787
```

Sample Output

```
0
1
1
```

Explanation

There are 4 events to process:

1. The first one sets the password to "000A".
2. The second one tries to authorize with the hash value 108738450. This value (which is the hash of the string "000B") doesn't correspond to the current password, nor to the current password with a single character appended to it. Therefore, this event returns 0.
3. The third event tries to authorize with the hash value 108738449. This is indeed the hash value of the current password, so this event returns 1.
4. Finally, the last event tries to authorize with hash value 244736787. This is the hash value of string "000AB", which is valid because it is equal to the current password with a single character appended to it. Therefore, this event returns 1.

▼ Sample Case 1

Sample Input

```
5
2
setPassword 1
setPassword 2
setPassword 3
authorize 49
authorize 50
```

Sample Output

```
0
0
```

Explanation

There are 5 events to process:

1. The first one sets the password to "1".
2. The second one sets the password to "2".
3. The third one sets the password to "3".
4. The fourth event tries to authorize with the hash value 49, which corresponds to "1". Because this is invalid for the current password of "3", this event returns 0.
5. The fifth event tries to authorize with the hash value 50, which corresponds to "2". Because this is invalid for the current password of "3", this event returns 0.

INTERVIEWER GUIDELINES

Editorial (pawel):

For each setPassword event, compute 62 different hashes. 10 for any digit appended to its end, 26 for any lowercase letter appended to its end, and other 26 for any uppercase letter appended to its end. Then, for authorize event, convert the given hash to int and compare it to these 62 hashes. Return 1 if any matches and 0 otherwise.

Setters' solution (pawel):

```
import string

P = 131
MOD = 10**9+7
VALID_CHARS = string.ascii_lowercase + string.ascii_uppercase +
string.digits
SET = "setPassword"
AUTH = "authorize"

def h(s):
    y = 0
    for c in s:
       y = (P*y + ord(c)) % MOD
    return y

def get_hashes(p):
    hashes = set([h(p)])
    for c in VALID_CHARS:
        hashes.add(h(p+c))
    return hashes

def authEvents(events):
    hashes = None
    res = []
    for event_type, param in events:
        if event_type == SET:
            hashes = get_hashes(param)
        else:
            param = int(param)
            res.append(int(param in hashes))
    return res
```

Tester's code:

```
def go_hash(word):
    res = 0
    for i in range(len(word)):
        res *= 131
        res += ord(word[i])
        res %= 1000000000 + 7
    return res

def authEvents(events):
    q = len(events)
    assert(q >= 2 and q <= 100000)
    password = ""
    res = []
    for i in range(q):
        if (events[i][0] == "setPassword"):
            password = events[i][1]
            assert(len(password) <= 9)
        else:
            words = set()
            words.add(go_hash(password))
            for j in range(48, 58):
                words.add(go_hash(password + chr(j)))
            for j in range(65, 91):
                words.add(go_hash(password + chr(j)))
            for j in range(97, 123):
                words.add(go_hash(password + chr(j)))
            assert(int(events[i][1]) < 1000000000 + 7 and int(events[i]
[1]) >= 0)
            if (int(events[i][1]) in words):
                res.append(1)
            else:
                res.append(0)
    return res
```

CANDIDATE ANSWER

Language used: **Python 3**

```

#
# Complete the 'authEvents' function below.
#
# The function is expected to return an INTEGER_ARRAY.
# The function accepts 2D_STRING_ARRAY events as parameter.
#
M = int(1e9 + 7)
p = 131

def handleSetPassword(password):
    currentHashValue = 0
    for i in range(len(password) - 1, -1, -1):
        currentHashValue = (currentHashValue + ((ord(password[i]) * p**
(len(password) - 1 - i)) %M )) % M
    return currentHashValue

def isHashValid(currentHash,newHash):
    if(newHash == currentHash):
        return True
    newHashBase = (currentHash * p) % M
    if(newHash < newHashBase):
        return M - newHashBase + newHash <= 127
    else:
        return newHash - newHashBase <= 127

def authEvents(events):
    res = []
    currentHash = None
    for event in events:
        if(event[0] == 'setPassword'):
            currentHash = handleSetPassword(event[1])
            #print(currentHash)
        else:
            res.append(1 if isHashValid(currentHash, int(event[1])) else 0)

    return res
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
TestCase 0	Easy	Sample case	✓ Success	1	0.0201 sec	10.6 KB
TestCase 1	Easy	Sample case	✓ Success	1	0.0194 sec	10.7 KB
TestCase 2	Easy	Sample case	✓ Success	1	0.12 sec	10.7 KB
TestCase 3	Easy	Hidden case	✓ Success	6	0.0221 sec	10.8 KB
TestCase 4	Easy	Hidden case	✓ Success	6	0.0188 sec	10.9 KB
TestCase 5	Easy	Hidden case	✓ Success	6	0.2169 sec	42.8 KB
TestCase 6	Easy	Hidden case	✓ Success	6	0.2201 sec	42.4 KB
TestCase 7	Easy	Hidden case	✓ Success	6	0.2302 sec	42.7 KB
TestCase 8	Easy	Hidden case	✓ Success	6	0.2201 sec	42.5 KB
TestCase 9	Easy	Hidden case	✓ Success	6	0.2252 sec	42.6 KB
TestCase 10	Easy	Hidden case	✓ Success	6	0.2718 sec	42.8 KB
TestCase 11	Easy	Hidden case	✓ Success	6	0.2207 sec	42.6 KB
TestCase 12	Easy	Hidden case	✓ Success	6	0.2299 sec	42.6 KB
TestCase 13	Easy	Hidden case	✓ Success	6	0.2241 sec	42.8 KB
TestCase 14	Easy	Hidden case	✓ Success	6	0.3125 sec	42.5 KB

No Comments

QUESTION 4

✓

Correct Answer

Score 75

Profit Targets › Coding

Binary Search | Data Structures | Medium | Algorithms | Arrays | Problem Solving | Theme: Finance | Interviewer Guidelines

QUESTION DESCRIPTION

A financial analyst is responsible for a portfolio of profitable stocks represented in an array. Each item in the array represents the yearly profit of a corresponding stock. The analyst gathers all distinct pairs of stocks that reached the target profit. Distinct pairs are pairs that differ in at least one element. Given the array of profits, find the number of distinct pairs of stocks where the sum of each pair's profits is exactly equal to the target profit.

Example

stocksProfit = [5, 7, 9, 13, 11, 6, 6, 3, 3]

target = 12 profit's target

- There are *4* pairs of stocks that have the sum of their profits equals to the target 12 . Note that because there are two instances of *3* in *stocksProfit* there are two pairs matching (9, 3): *stocksProfits* indices 2 and 7, and indices 2 and 8, but only one can be included.
- There are *3* distinct pairs of stocks: *(5, 7)*, *(3, 9)*, and *(6, 6)* and the return value is *3*.

Function Description

Complete the function *stockPairs* in the editor below.

stockPairs has the following parameter(s):

int stocksProfit[n]: an array of integers representing the stocks profits
target: an integer representing the yearly target profit

Returns:

int: the total number of pairs determined

Constraints

- $1 \le n \le 5 \times 10^5$
- $0 \le stocksProfit[i] \le 10^9$
- $0 \le target \le 5 \times 10^9$

▼ Input Format for Custom Testing

Input from stdin will be processed as follows and passed to the function.

The first line contains an integer *n*, the size of the array *stocksProfit*.
The next *n* lines each contain an element *stocksProfit[i]* where *0 ≤ i < n.*
The next line contains an integer *target*, the target value.

▼ Sample Case 0

Sample Input 0

```
STDIN      Function
-----      --------
6      →   stocksProfit[] size n = 6
1      →   stocksProfit = [1, 3, 46, 1, 3, 9]
3
46
1
3
9
47     →   target = 47
```

Sample Output 0

```
1
```

Explanation 0

There are *4* pairs where *stocksProfit[i] + stocksProfit[j]* = 47

1. *(stocksProfit0] = 1, stocksProfit[2] = 46)*
2. *(stocksProfit[2] = 46, stocksProfit[0] = 1)*
3. *(stocksProfit[2] = 46, stocksProfit[3] = 1)*
4. *(stocksProfit[3] = 1, stocksProfit[2] = 46)*

Since all four pairs contain the same values, there is only *1 distinct* pair of stocks : *(1, 46)*.

▼ Sample Case 1

Sample Input 1

```
STDIN      Function
-----      --------
7      →   stocksProfit[] size n = 7
6      →   stocksProfit = [6, 6, 3, 9, 3, 5, 1]
6
3
9
3
5
1
12     →   target = 12
```

Sample Output 1

```
2
```

Explanation 1

There are *5* pairs where *stocksProfit[i] + stocksProfit[j]* = 12:

1. *(stocksProfit[0] = 6, stocksProfit[1] = 6)*
2. *(stocksProfit[1] = 6, stocksProfit[0] = 6)*
3. *(stocksProfit[2] = 3, stocksProfit[3] = 9)*
4. *(stocksProfit[3] = 9, stocksProfit[2] = 3)*
5. *(stocksProfit[3] = 9, stocksProfit[4] = 3)*
6. *(stocksProfit[4] = 3, stocksProfit[3] = 9)*

The first *2* pairs are the same, as are the last *4*. There are only *2 distinct* pairs of stocks: *(3, 9)* and *(6, 6)*.

INTERVIEWER GUIDELINES

▼ Hint 1

Is there an efficient way you can find out whether target - stocksProfit[i] exists in the array for every i?

▼ Hint 2

Multiple occurrences of the same value don't contribute to the final answer except in one special case, target/2 when target is even. Try using hash tables.

▼ Solution

Concepts covered: Hash Table

Optimal Solution:

Suppose that we already know the value of the first stock, call it *value*. We can say that the value of the second stock must be target - value. Then we just need to find out whether target - value exists in the array. We can to this efficiently using a hash table. One point to notice here is that if target is divisible by 2, then there must be at least two occurrences of target/2 in the array for it to contribute in the final answer.

```
def stockPairs(stocksProfit, target):
    stock_values = set(stocksProfit)
    ans = 0
    for value in stock_values:
        if target - value in stock_values and target != 2 * value:
            ans += 1
    if target % 2 == 0 and stocksProfit.count(target // 2) > 1:
        ans += 2
    return ans // 2
```

Brute Force Approach: Passes 13 of 15 test cases

```
def stockPairs(stocksProfit, target):
    values_taken = set()
    ans = 0
    n = len(stocksProfit)
    for i in range(n):
        for j in range(i+1, n):
            if stocksProfit[i] + stocksProfit[j] == target and
(min(stocksProfit[i], stocksProfit[j]), max(stocksProfit[i],
stocksProfit[j])) not in values_taken:
                ans += 1
                values_taken.add((min(stocksProfit[i], stocksProfit[j]),
max(stocksProfit[i], stocksProfit[j])))
    return ans
```

Error Handling: The edge case which candidates must take care is when target is divisible by 2 and the number of occurrences of target/2 is equal to 1.

▼ Complexity Analysis

Time Complexity - O(n).

Since we are iterating over each element exactly once and for each element we are doing a lookup in the hash table (O(1) time complexity), each pass costs O(1) time. The overall time complexity is O(n).

Space Complexity - O(n).

The hash table takes O(n) space.

▼ Follow up Question

What if the task is to find out the number of distinct pair of stocks such that their sum is ≥ target?

Now, for each element value we need to query the number of integers which are ≥ target - value. This can be done using a binary search tree.

CANDIDATE ANSWER

Language used: **Python 3**

```

#
# Complete the 'stockPairs' function below.
#
# The function is expected to return an INTEGER.
# The function accepts following parameters:
#  1. INTEGER_ARRAY stocksProfit
#  2. LONG_INTEGER target
#
from collections import defaultdict
def stockPairs(stocksProfit, target):
    # Write your code here
    stocksProfitCopy = stocksProfit[:]
    stocksProfitCopy.sort()
    print(stocksProfitCopy)
    left = 0
    right = len(stocksProfitCopy) - 1
    foundPairs = defaultdict(lambda: set())
    res = 0
    while(right > left):
        while(right > left and stocksProfitCopy[left] +
stocksProfitCopy[right] > target):
            right -= 1
        while(right > left and stocksProfitCopy[left] +
stocksProfitCopy[right] < target):
            left += 1
        if(right != left and stocksProfitCopy[right] + stocksProfitCopy[left]
== target):
            if(not stocksProfitCopy[right] in
foundPairs[stocksProfitCopy[left]]):

foundPairs[stocksProfitCopy[left]].add(stocksProfitCopy[right])
                res += 1
            left += 1
    return res
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0: O(n^2)	Easy	Sample case	✓ Success	1	0.0199 sec	10.8 KB
Testcase 1: O(n^2) | Edge Case : 2 occurences of element with value target/2	Easy	Sample case	✓ Success	1	0.0195 sec	10.8 KB
Testcase 2: O(n^2) | Edge Case : 1 occurence of element with value target/2	Easy	Sample case	✓ Success	1	0.027 sec	10.6 KB
Testcase 3: O(n^2)	Easy	Hidden case	✓ Success	2	0.0191 sec	10.7 KB
Testcase 4: O(n^2)	Easy	Hidden case	✓ Success	2	0.0214 sec	10.7 KB
Testcase 5: O(n^2)	Easy	Sample case	✓ Success	2	0.0257 sec	10.7 KB
Testcase 6: O(n^2)	Easy	Hidden case	✓ Success	2	0.0207 sec	10.9 KB
Testcase 7: O(n^2)	Easy	Hidden case	✓ Success	2	0.0204 sec	10.8 KB
Testcase 8: O(n^2)	Medium	Hidden case	✓ Success	4	0.0246 sec	11.1 KB
Testcase 9: O(n^2)	Medium	Hidden case	✓ Success	4	0.0227 sec	10.9 KB
Testcase 10: O(n^2)	Medium	Hidden case	✓ Success	5	0.0302 sec	11.3 KB
Testcase 11: O(n^2)	Medium	Sample case	✓ Success	5	0.0363 sec	11.4 KB
Testcase 12: O(n^2)	Medium	Hidden case	✓ Success	6	0.0275 sec	11.3 KB
Testcase 13: O(n) or O(n logn)	Hard	Hidden case	✓ Success	19	0.1323 sec	17.2 KB
Testcase 14: O(n) or O(n logn)	Hard	Hidden case	✓ Success	19	0.2458 sec	24.5 KB

No Comments

QUESTION 5

! Needs Review

Score 75

JavaScript: Activity List › Coding

Medium | Javascript | Coding | Prototypal inheritance

QUESTION DESCRIPTION

The goal of this problem is to use prototypal inheritance in Javascript.

Implement inheritance as described below-

Create a function *Activity* that takes a single parameter *amount* (Number) and assigns it to member variable '*amount*'.
Add the following functions to the *Activity* prototype -

1. *setAmount* - This function takes a single parameter , *value*.
   - If the value is less than or equal to 0, it returns false.
   - Otherwise, it assigns *value* to the member variable *amount* and returns true.
2. *getAmount* - This function returns the member variable *amount* value.

Create a function *Payment* that -

1. inherits from parent *Activity*.
2. takes 2 parameters - *amount* (Number) and *receiver* (string). It assigns the parent's member variable '*amount*', and self's member variable '*receiver*' respectively.

Add the following functions to *Payment's* existing prototype -

1. *setReceiver* - This function takes a single parameter and assigns it to the member variable '*receiver*'.
2. *getReceiver* - This function returns the member variable '*receiver*' value.

Create a function *Refund* that -

1. inherits from parent *Activity*.
2. takes 2 parameters - *amount* (Number) and *sender* (string) and assigns the parent's member variable, '*amount*', and self's member variable, '*sender*'.

Add below functions to *Refund's* existing prototype -

1. *setSender* - This function takes a single parameter and assigns it to the member variable *sender*.
2. *getSender* - This function returns the member variable *sender*.

Implementation of the function will be tested by the provided code stub using several input files. Each input file contains parameters for the function calls. The result of their executions will be printed to the standard output by the provided code. In the case of a *setAmount* function call, if the value returned is false, the stubbed code prints 'Amount not updated'. If the value returned is true, it prints 'Amount updates to <value>'.

▼ Input Format For Custom Testing

The first line specifies the function name for which object needs to be created i.e. either *Payment* or *Refund*.

The second line contains space-separated initial amount and receiver/sender values for initial object creation.

The third line contains an integer to update the amount.

The fourth line contains a string to update the receiver/sender value of the object.

▼ Sample Case 0

Sample Input For Custom Testing

```
STDIN         Function
-----         --------
Payment    rarr    object to create = Payment
5000 John    rarr    initial amount = 5000, initial receiver = 'John'
4000        rarr    update amount to 4000
John B    rarr    update receiver to 'John B'
```

Sample Output

```
Payment object created with amount 5000 and receiver John
Amount updated to 4000
Receiver updated to John B
Payment object details - amount is 4000 and receiver is John B
Payment.prototype has property setAmount: false
Payment.prototype has property getAmount: false
Payment.prototype has property setReceiver: true
Payment.prototype has property getReceiver: true
```

Explanation

A Payment object is created with the *amount* as 5000 and *receiver* as 'John' (inputs from the second line). The third and fourth lines have updated *amount* and *receiver* values that are used to update the object's member variables using *setAmount* and *setReceiver* functions. Since *setAmount* returns true in this case, it prints 'Amount updated to 4000'. The stub code then calls the *getAmount* and *getReceiver* functions and prints the values. The last 4 lines check if the object prototype chain is built correctly.

▼ Sample Case 1

Sample Input For Custom Testing

```
Refund
5000 John
4000
John B
```

Sample Output

```
Refund object created with amount 5000 and sender John
Amount updated to 4000
Sender updated to John B
Refund object details - amount is 4000 and sender is John B
Refund.prototype has property setAmount: false
Refund.prototype has property getAmount: false
Refund.prototype has property setSender: true
Refund.prototype has property getSender: true
```

Explanation

A Refund object is created with the *amount* as 5000 and *sender* as 'John' (inputs from the second line). The third and fourth lines have updated *amount* and *sender* values that are used to update the object's member variables using *setAmount* and *setSender* functions. Since *setAmount* returns true in this case, it prints 'Amount updated to 4000'. The stub code then calls get*Amount* and *getSender* functions and prints the values. The last 4 lines check if the object prototype chain is built correctly.

CANDIDATE ANSWER

Language used: **JavaScript (Node.js)**

```javascript

function Activity(amount) {
    this.amount = amount;
}
Activity.prototype.getAmount = function(){
    return this.amount;
}
Activity.prototype.setAmount = function(value){
    if(value <= 0){
        return false;
    }
    this.amount = value;
    return true;
}

function Payment(amount, receiver) {
    Activity.call(this, amount);
    this.receiver = receiver;
}
Payment.prototype = Object.create(Activity.prototype);
Payment.prototype.setReceiver = function(receiver){this.receiver = receiver}
Payment.prototype.getReceiver = function(){return this.receiver}

function Refund(amount, sender) {
    Activity.call(this, amount);
    this.sender = sender;
}
Refund.prototype = Object.create(Activity.prototype)
Refund.prototype.setSender = function(sender){this.sender = sender;}
Refund.prototype.getSender = function(){return this.sender}

```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	✓ Success	1	0.0388 sec	42.1 KB
Testcase 1	Easy	Sample case	✓ Success	1	0.0532 sec	42 KB
Testcase 2	Medium	Hidden case	✓ Success	10	0.035 sec	42 KB
Testcase 3	Medium	Hidden case	✓ Success	10	0.0347 sec	41.9 KB

Testcase 4	Medium	Hidden case	✓ Success	10	0.0456 sec	41.9 KB
Testcase 5	Medium	Hidden case	✓ Success	10	0.0345 sec	41.9 KB
Testcase 6	Medium	Hidden case	✓ Success	10	0.0501 sec	41.9 KB
Testcase 7	Medium	Hidden case	✓ Success	10	0.0429 sec	42 KB
Testcase 8	Easy	Hidden case	✓ Success	7	0.0425 sec	41.9 KB
Testcase 9	Easy	Hidden case	✓ Success	6	0.041 sec	42.1 KB

No Comments

PDF generated at: 6 Dec 2023 05:18:44 UTC