**Problem Statement: Secure Storage Access Control**

You are designing a secure storage system for a company. The storage system uses a hashset to manage access controls. Each user can be granted access to specific files, and the system must efficiently handle operations to add, remove, and check access permissions.

The system should support the following operations:

1. grantAccess user file: Grant the specified user access to the specified file.
2. revokeAccess user file: Revoke the specified user's access to the specified file.
3. checkAccess user file: Check if the specified user has access to the specified file.

**Scenario:** Consider a scenario where multiple users need access to different files in a secure storage system. For example, engineers may need access to design documents, while managers may need access to project plans. The system should ensure that access permissions can be granted, revoked, and checked efficiently.

**Input Format:**

1. The first line contains an integer n denoting the number of operations.
2. The next n lines contain operations in the format:
   * grantAccess user file
   * revokeAccess user file
   * checkAccess user file

**Output Format:** For each checkAccess operation, print "Granted" if the user has access to the file. Otherwise, print "Denied".

**Constraints:**

* The number of operations n is in the range [1, 10000].
* user and file are strings consisting of lowercase English letters and digits, and their lengths are in the range [1, 100].
* Each user-file pair is unique in the context of access control.

**Sample Input:**

6

grantAccess alice design\_doc

grantAccess bob project\_plan

checkAccess alice design\_doc

revokeAccess alice design\_doc

checkAccess alice design\_doc

checkAccess bob project\_plan

**Sample Output:**

Granted

Denied

Granted

**Explanation:**

* The first operation grants alice access to design\_doc.
* The second operation grants bob access to project\_plan.
* The third operation checks if alice has access to design\_doc, which is granted.
* The fourth operation revokes alice's access to design\_doc.
* The fifth operation checks if alice has access to design\_doc, which is denied after revocation.
* The sixth operation checks if bob has access to project\_plan, which is granted.

**Hints:**

1. Use a hashset to store and manage the access permissions.
2. For the checkAccess operation, simply check if the user-file pair exists in the hashset.

**Solution Template:**

python

class SecureStorage:

def \_\_init\_\_(self):

self.access\_set = set()

def grantAccess(self, user: str, file: str) -> None:

self.access\_set.add((user, file))

def revokeAccess(self, user: str, file: str) -> None:

self.access\_set.discard((user, file))

def checkAccess(self, user: str, file: str) -> str:

if (user, file) in self.access\_set:

return "Granted"

else:

return "Denied"

def main():

import sys

input = sys.stdin.read

data = input().strip().split('\n')

n = int(data[0])

storage = SecureStorage()

results = []

for i in range(1, n + 1):

command = data[i].split()

operation = command[0]

user = command[1]

file = command[2]

if operation == "grantAccess":

storage.grantAccess(user, file)

elif operation == "revokeAccess":

storage.revokeAccess(user, file)

elif operation == "checkAccess":

result = storage.checkAccess(user, file)

results.append(result)

for result in results:

print(result)

if \_\_name\_\_ == "\_\_main\_\_":

main()

**Extra Test Cases:**

**Test Case 1:**

7

grantAccess john report

grantAccess jane analysis

checkAccess john report

checkAccess jane report

revokeAccess john report

checkAccess john report

checkAccess jane analysis

**Output:**

Granted

Denied

Denied

Granted

**Test Case 2:**

5

grantAccess alice summary

grantAccess bob details

revokeAccess alice summary

checkAccess alice summary

checkAccess bob details

**Output:**

Denied

Granted

**Test Case 3:**

8

grantAccess tom design

grantAccess tom plan

grantAccess jerry blueprint

revokeAccess tom plan

checkAccess tom design

checkAccess tom plan

checkAccess jerry blueprint

revokeAccess jerry blueprint

**Output:**

Granted

Denied

Granted

**Test Case 4:**

6

grantAccess user1 file1

grantAccess user1 file2

grantAccess user2 file1

revokeAccess user1 file1

checkAccess user1 file1

checkAccess user2 file1

**Output:**

Denied

Granted

**Test Case 5:**

9

grantAccess manager1 confidential

grantAccess manager2 confidential

grantAccess employee1 public

grantAccess employee2 public

checkAccess manager1 confidential

checkAccess employee1 confidential

checkAccess employee1 public

revokeAccess manager1 confidential

checkAccess manager1 confidential

**Output:**

Granted

Denied

Granted

Denied