```
import requests
import json
import sys
from datetime import datetime

# Firebase Database configuration file
CONFIG_FILE = "config.json"

# 1. Send a chat message
def send_chat_message(sender, receiver, message):
    # INPUT : Sender, Receiver, Message Body
    # RETURN : Status code of the Firebase REST API call [response.status_code]
    # EXPECTED RETURN : 200

    # Load config as dict object
    config = json.load(open(CONFIG_FILE))

    # Get current timestamp
    time = int(datetime.now().timestamp())

    # Store only the recent message between sender and receiver under both sender's and receiver's
names
    # independent of sender and receiver roles for extacting the recent messages of a person in
function 2

        # Storing the most recent message for the sender, replacing the previous message when new one
comes
    sender_id_url = config['dburl'] + f'{config["node"]}/individual_id/{sender}.json'
    response1 = requests.put(sender_id_url, json={"sender" : sender, "receiver" : receiver, "body" :
message, "timestamp" : time})

        # Storing the most recent message for the receiver, replacing the previous message when new
one comes
    receiver_id_url = config['dburl'] + f'{config["node"]}/individual_id/{receiver}.json'
    response2 = requests.put(receiver_id_url, json={"sender" : sender, "receiver" : receiver, "body"
: message, "timestamp" : time})

    # Storing timestamped message under conversation_id for retrieving last k messages between two
people
    # Since the roles of sender and receiver are not required for extracting the messages in the
conversation, the
    # messages are stored under common conversation_id
    conversation_id_url = config['dburl'] + f'{config["node"]}/conversation_id/{"-
".join(sorted([sender, receiver]))}/{time}.json'
    response3 = requests.put(conversation_id_url, json={"sender" : sender, "receiver" : receiver,
"body" : message})

    if response1.status_code == response2.status_code == response3.status_code == 200:
        return 200
    return



# 2. Retrieve the most recent message for a person
def get_recent_message(person):
    # INPUT : Person (as sender or receiver)
    # RETURN : JSON object with details of the most recent message or None if no message exists
    # EXPECTED RETURN : {"sender": "john", "receiver": "david", "body": "hello", "timestamp":
1674539458} or None
    # Load config as dict object
    config = json.load(open(CONFIG_FILE))

    # Query for recent chat of person (irrespective of sender or receiver roles) from individual
index
```

```
    person_id_url = config['dburl'] + f'{config["node"]}/individual_id/{person}.json?print=pretty'
    recent_chat = requests.get(person_id_url)

    if recent_chat.content != b'null':
        return recent_chat.json()
    else:
        return


# 3. Retrieve the last k messages between two people
def get_last_k_messages(person1, person2, k):
    # INPUT : Person1, Person2, Number of messages (k)
    # RETURN : List of JSON objects with the k most recent messages or an Empty list if no messages
exist
    # EXPECTED RETURN : [{"sender": "john", "receiver": "david", "body": "hello", "timestamp":
1674539458}, ...] or []
    # Load config as dict object
    config = json.load(open(CONFIG_FILE))

    # Query for recent k chats between person1 and person2 (irrespective of sender or receiver roles)
from conversation index
    conversation_id_url = config['dburl'] + f'{config["node"]}/conversation_id/{"-
".join(sorted([person1, person2]))}.json?orderBy="$key"&limitToLast={k}&print=pretty'
    recent_chats = requests.get(conversation_id_url)

    result  = []
    if recent_chats.content != b'null':
        recent_chats_dict = recent_chats.json()
        # format the topk chats such that the older messages appear first
        ts_ordered_chats = sorted(list(recent_chats_dict.items()), key=lambda x: x[0])
        for ts, chat_info in ts_ordered_chats:
            result += [{"sender" : chat_info['sender'], "receiver" : chat_info['receiver'], "body" :
chat_info['body'], "timestamp":ts}]
    return result


# Main execution logic
if __name__ == "__main__":
    if len(sys.argv) < 2:
        print("Usage: python solution_code_hw1.py [operation] [arguments]")
        sys.exit(1)

    command = sys.argv[1].lower()

    if command == "send_message":
        sender = sys.argv[2]
        receiver = sys.argv[3]
        message = sys.argv[4]
        result = send_chat_message(sender, receiver, message)
        print(result)

    elif command == "get_recent":
        person = sys.argv[2]
        result = get_recent_message(person)
        print(result)

    elif command == "get_last_k":
        person1 = sys.argv[2]
        person2 = sys.argv[3]
        k = int(sys.argv[4])
        result = get_last_k_messages(person1, person2, k)
        print(result)

    else:
        print("Invalid command. Use 'send_message', 'get_recent', or 'get_last_k'.")
```