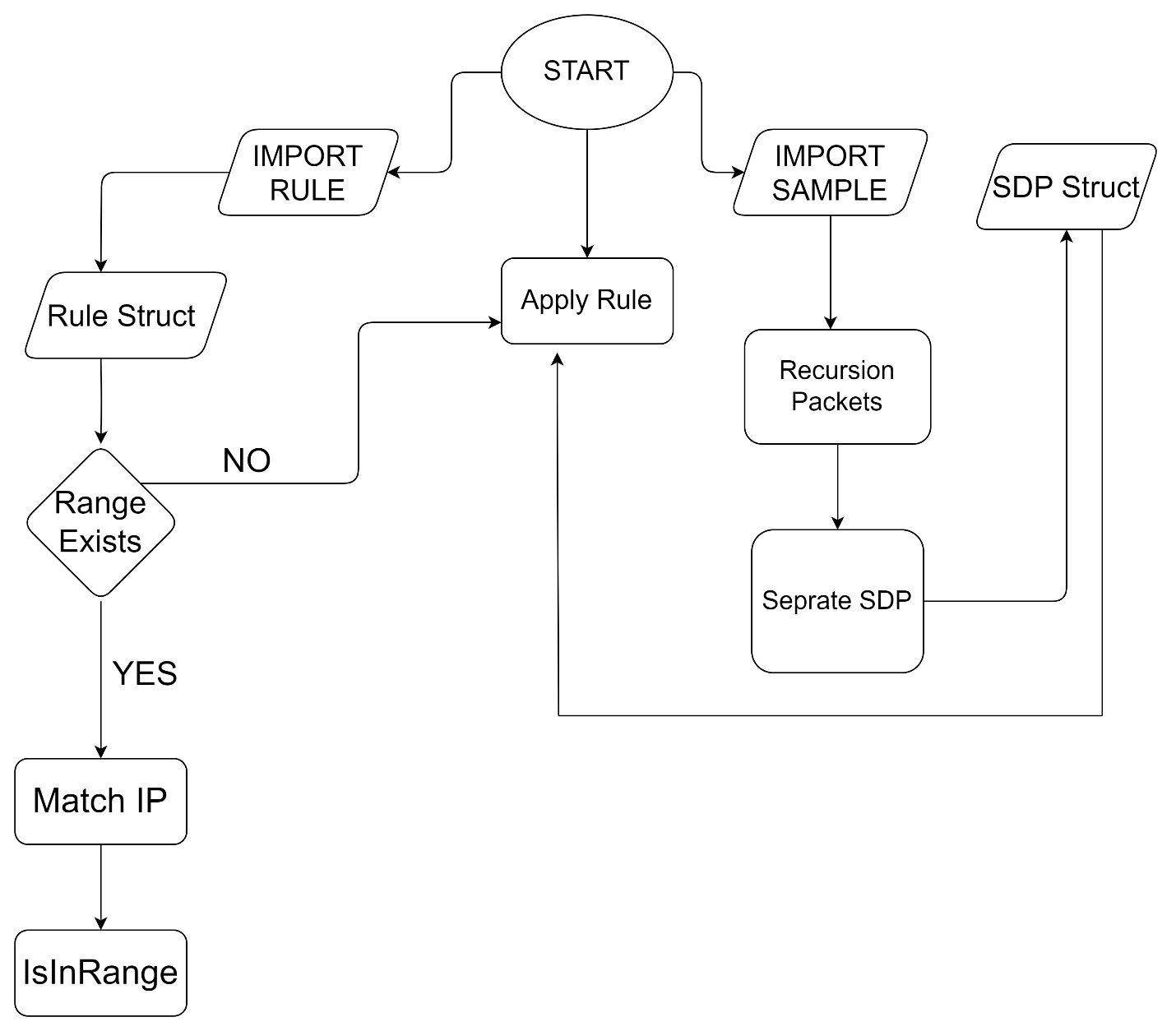
**FIREWALL**

**Flowchart:**

High level design showing flow of functions in the program.



**1. Code Overview**

The main purpose of the code is to read data about packets and rules from files. It then applies these rules to the packets and creates a result file showing which rules matched. To do this, the code uses two important structures: SDP for packets and Rules for storing rule details.

* **File Handling**

The code uses files to get information about packets and rules. It checks for errors to make sure everything works even if files are missing or not set up correctly. This helps the program handle file operations well.

* **Structures**

The SDP structure holds details about each packet like where it came from, where it's going and its protocol. On the other hand, the Rules structure contains information about each rule like the rule number where it applies, whether it's about IP or protocol and what action it takes.

**2. Implementation Details**

* **Packet Parsing**

The code breaks down packets from an input string using a recursive function in the Packets function. The Seprate function then takes this and pulls out details like source, destination, and protocol, filling in the SDP structure.

* **Rule Application**

The applyRule function checks each packet with the given rules. It has a way to deal with rules that specify ranges of IP addresses. The program then makes a result file showing how each rule applies to each packet.

* Top of Form
* **File Input/Output**

Working with files is really important for the system to work. The code reads information about packets and rules from files and then writes the results to another file. Doing things this way makes it simple for users to change the input files whenever they need to.

**3. Code Structure and Organization**

The code is organized in such a way to make it easy to read and maintain. Functions like Import\_Sample\_File, Import\_Rule\_File, and applyRule each do specific jobs, making the code clear and well organized.

* **Error Handling**

The program checks for errors to handle issues like missing files or files in the wrong format. While these checks make the program strong, there's room to make error handling even better with some improvements.

**4. Problems Faced.**

A) The variable (index\_of\_SDP\_struct) was initialized zero in the (Seprate) function where source , destination , protocol is storing. But it was not storing correctly because then it was not incrementing. Turns out this is due to the recursive function when because (Seprate) was called in recursive function index\_of\_SDP\_struct was again initialized to ‘0’, To overcome this problem we made it a global variable.

B) Second problem came in ranging where we were basically finding the ‘-‘ then converting its neighbors into integers, and then compare but that was a limitation because it could check the range in only the last octet. We overcame that by checking if the rule file has any range then its octets were broken and then it was compared octet by octet.

**5. Suggestions for Improvement**

* **Global Variables**

We could have used variables like index\_of\_SDP\_struct differently. It's good to use variables in the best way possible. Instead of using global variables, we could have directly sent the needed information to functions. This could have made the code easier to handle.

* **Recursion**

The way the Packets function uses recursion might be risky for very large inputs, as it could lead to a stack overflow. Changing the recursive approach to a loop could make the program handle large inputs better. But we have deliberately used the recursive function to get better understand of it and to make our code compact and smaller in size.

**5. Conclusion**

In summary, the Network Packet Rule Matching System is a helpful tool for network administrators. It allows them to analyze and respond to network traffic (Data Packets) using predefined rules. The documentation is meant to give a better understanding of what the code does, how it works and where it could be made better.

**6. Contributions**

**Ali Hamza :** Recursion , Seprating (src,dst,pro) comparing rules (comparing ranges)

**Basim Mehdi :** Storing src,dst,pro in a struct , Helped in logicbuilding , Project Documentation, Designed Algorithm

**Abdullah Mehmood :** Importing rule file and storing it in struct , Helped in logic building , Designed Algorithm

**Soban Yam :** Helped in logicbuilding , Designed Flowchart , Designed Algorithm

**It was all teamwork! ☺**