INFORMATION SECURITY

PRACTICAL LAB FILE

of

BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTING

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY UNA, HIMACHAL PRADESH

2020-2021

Submitted to
Prof. Bhuvaneswari Amma N.G.

Submitted by
Name: Pradeep Kumar
Roll: 17119

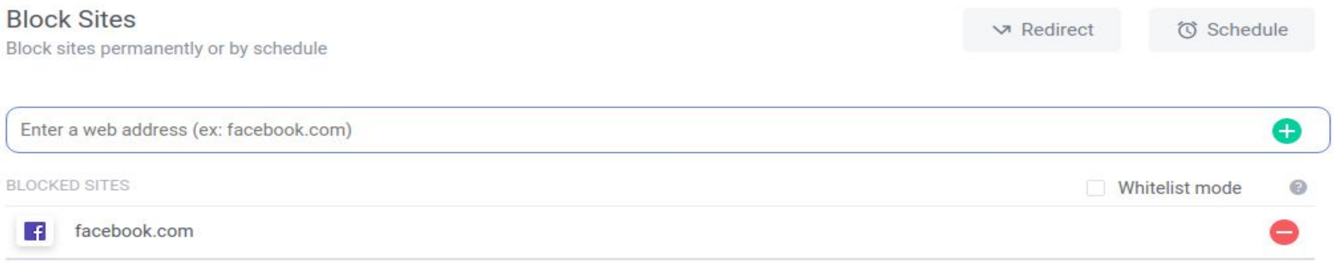
List of Experiments

PRACTICAL 1

Q. Select any browser and secure the browser by the following settings:

i) Trusted sites/blocked sites:

a. Search for the *Block Site* Chrome extension, and add it to browser.
b. Click *Add extension* in the pop-up box.
c. Check for the extension's icon on the top-right hand corner of your Chrome screen.
d. Visit a website you want to block from then on.



Block Sites
Block sites permanently or by schedule

BLOCKED SITES

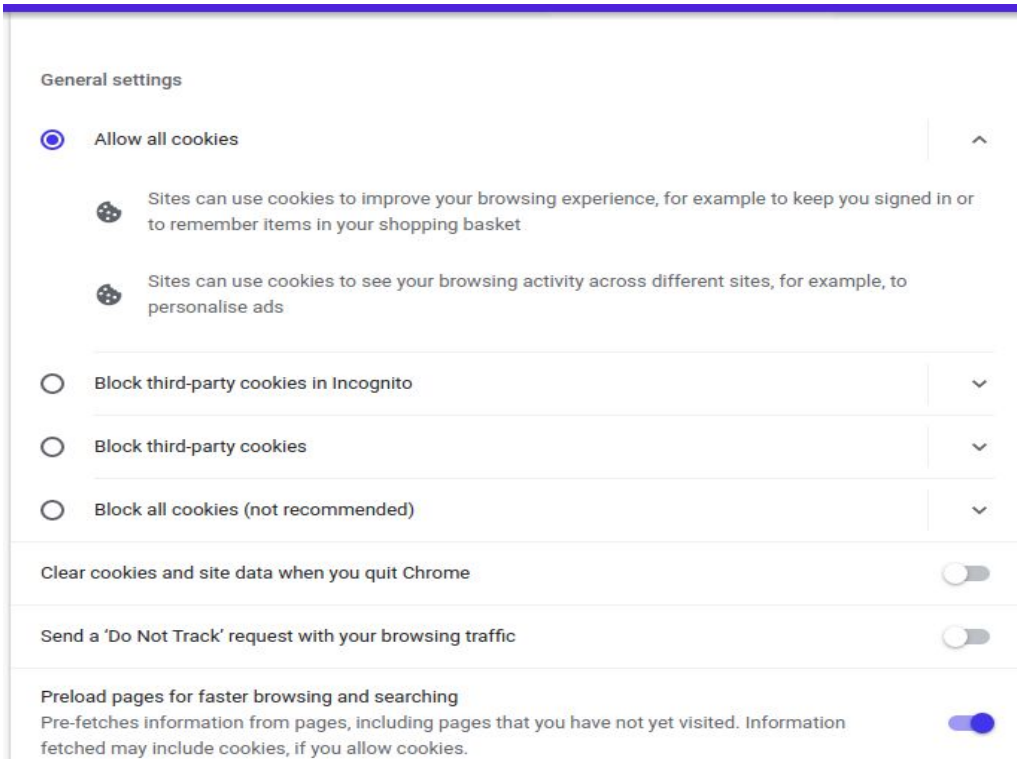
□ Whitelist mode

facebook.com

ii) Enabling or disabling cookies:

Open Google chrome, Go to **Settings > Privacy and Security > Cookies and Other Site Data**

We can Select any of the following option:



General settings

● Allow all cookies

Sites can use cookies to improve your browsing experience, for example to keep you signed in or to remember items in your shopping basket

Sites can use cookies to see your browsing activity across different sites, for example, to personalise ads

○ Block third-party cookies in Incognito

○ Block third-party cookies

○ Block all cookies (not recommended)

Clear cookies and site data when you quit Chrome

Send a 'Do Not Track' request with your browsing traffic

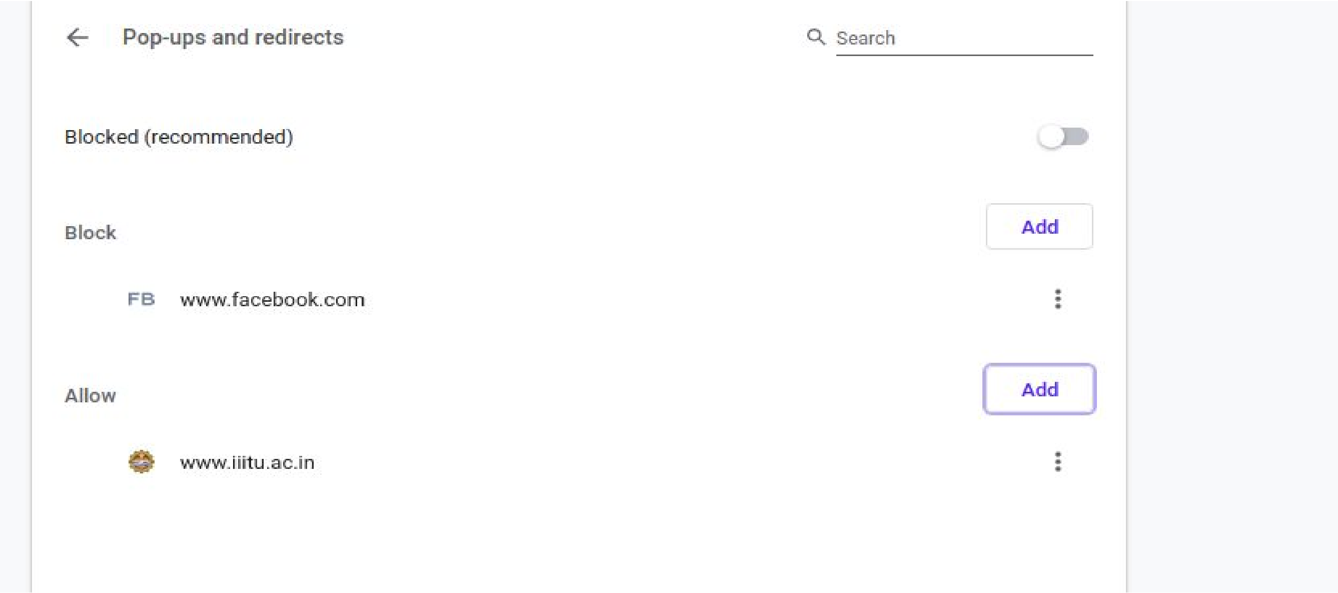
Preload pages for faster browsing and searching
Pre-fetches information from pages, including pages that you have not yet visited. Information fetched may include cookies, if you allow cookies.

iii) Use of pop up blocker

Pop-Up : Pop-up ads or pop-ups are forms of online advertising on the World Wide Web. A pop-up is a graphical user interface (GUI) display area, usually a small window, that suddenly appears ("pops up") in the foreground of the visual interface.

We can block pop-ups to have smooth web surfing.

Open Google chrome, Go to **Settings > Privacy and Security > Site Settings > Additional Permissions > Content > Pop-Up and Redirects**



Pop-ups and redirects

Blocked (recommended)

Block

FB www.facebook.com

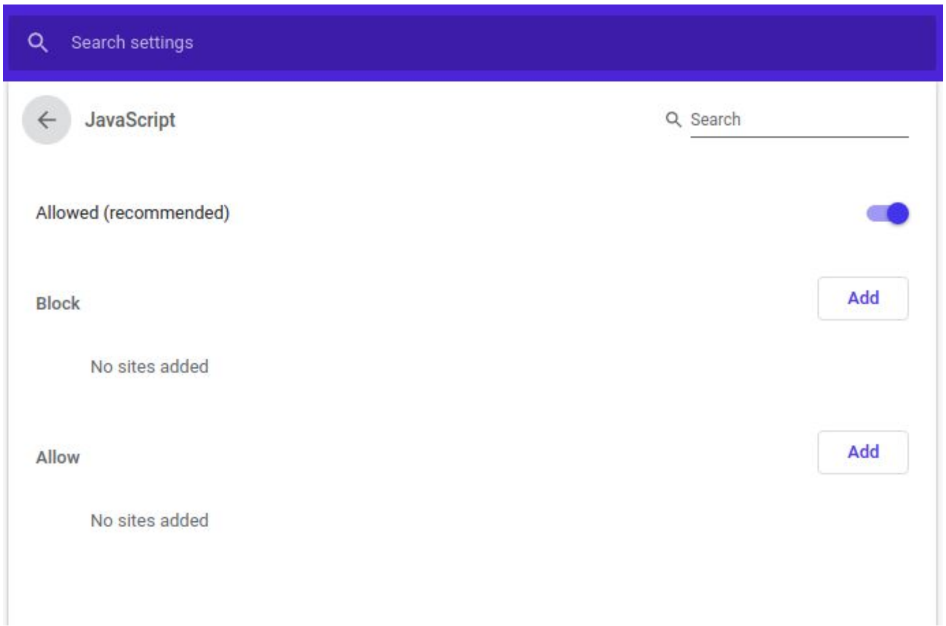
Allow

www.iiitu.ac.in

We can allow and block Pop-Ups and Redirects for certain websites. We can add these websites just by clicking on “Add”.

iv) Enabling or disabling scripts

Settings > Privacy and Security > Site Settings > Additional Permissions > Content > JavaScript



JavaScript

Allowed (recommended)

Block

No sites added

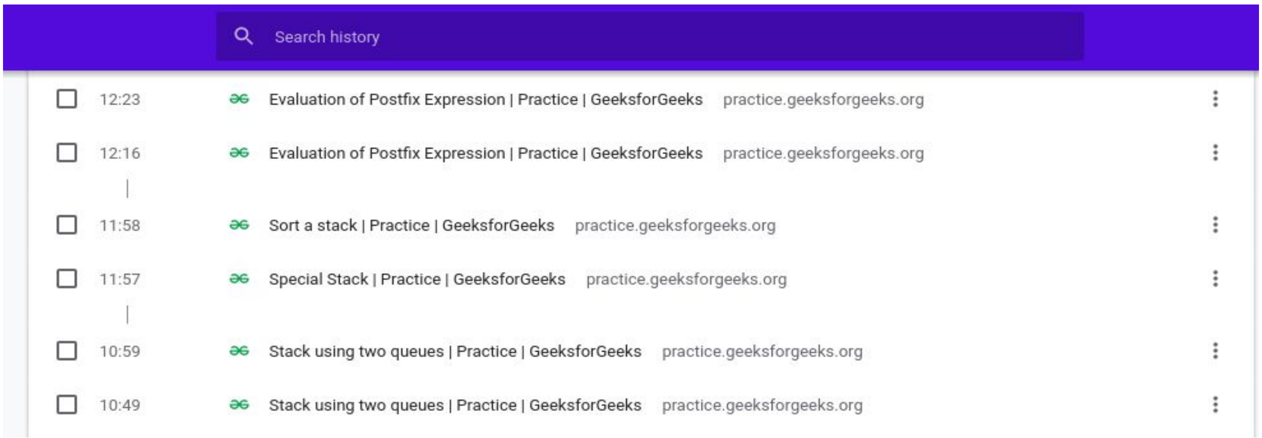
Allow

No sites added

We can allow and block JavaScripts for certain websites. We can add these websites just by clicking on “Add”.

v) Browsing history

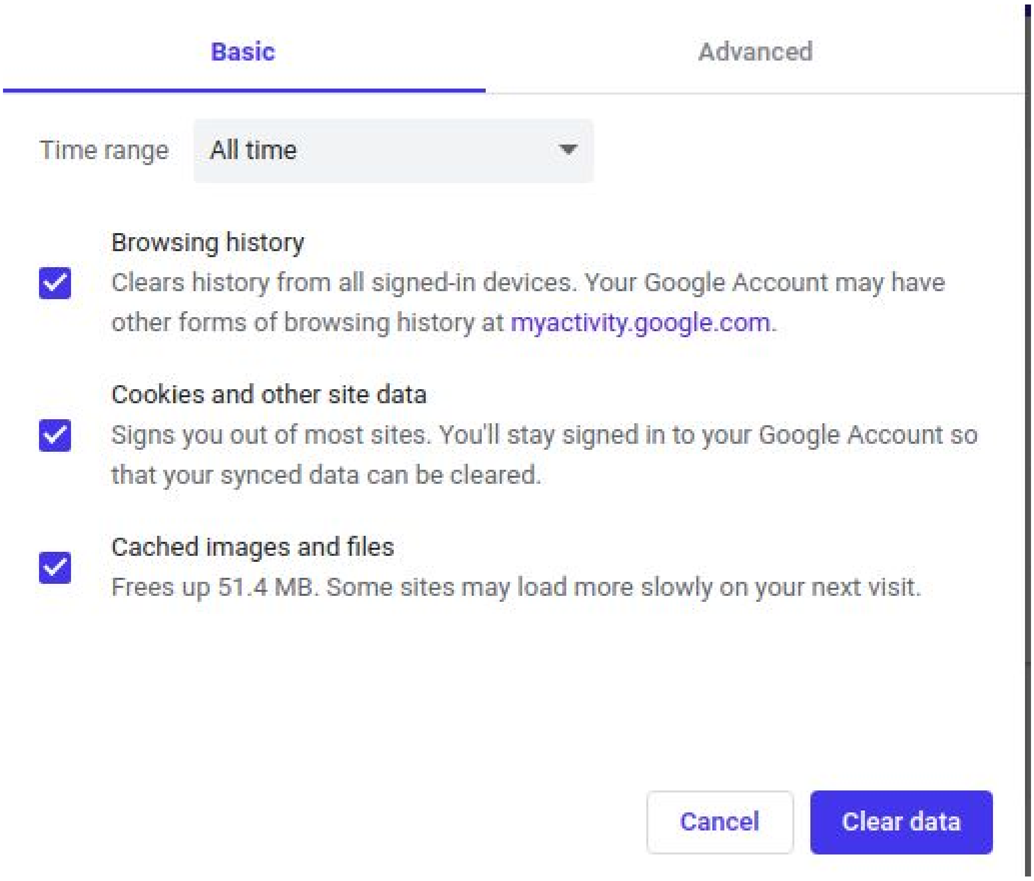
The keyboard short-cut to access Browsing history is “**Ctrl + H**”.



□	12:23	Evaluation of Postfix Expression | Practice | GeeksforGeeks	practice.geeksforgeeks.org
□	12:16	Evaluation of Postfix Expression | Practice | GeeksforGeeks	practice.geeksforgeeks.org
□	11:58	Sort a stack | Practice | GeeksforGeeks	practice.geeksforgeeks.org
□	11:57	Special Stack | Practice | GeeksforGeeks	practice.geeksforgeeks.org
□	10:59	Stack using two queues | Practice | GeeksforGeeks	practice.geeksforgeeks.org
□	10:49	Stack using two queues | Practice | GeeksforGeeks	practice.geeksforgeeks.org

We can delete the browsing History by clicking on “**Clear Browsing Data”** from the Side-Nav-Bar

a.) Basic Options



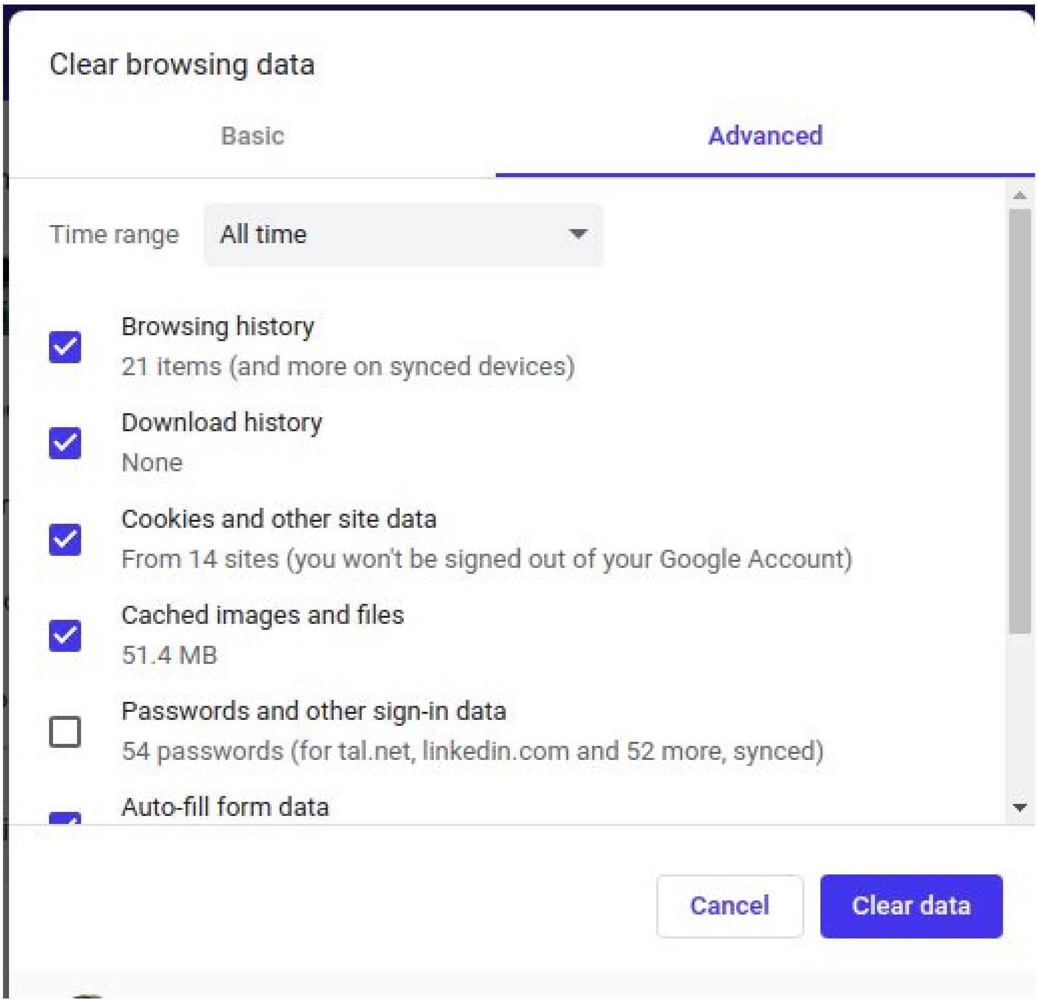
Time range: All time

☑ Browsing history
Clears history from all signed-in devices. Your Google Account may have other forms of browsing history at myactivity.google.com.

☑ Cookies and other site data
Signs you out of most sites. You'll stay signed in to your Google Account so that your synced data can be cleared.

☑ Cached images and files
Frees up 51.4 MB. Some sites may load more slowly on your next visit.

b.) Advanced Options



Clear browsing data

Basic | Advanced

Time range: All time

☑ Browsing history
21 items (and more on synced devices)

☑ Download history
None

☑ Cookies and other site data
From 14 sites (you won't be signed out of your Google Account)

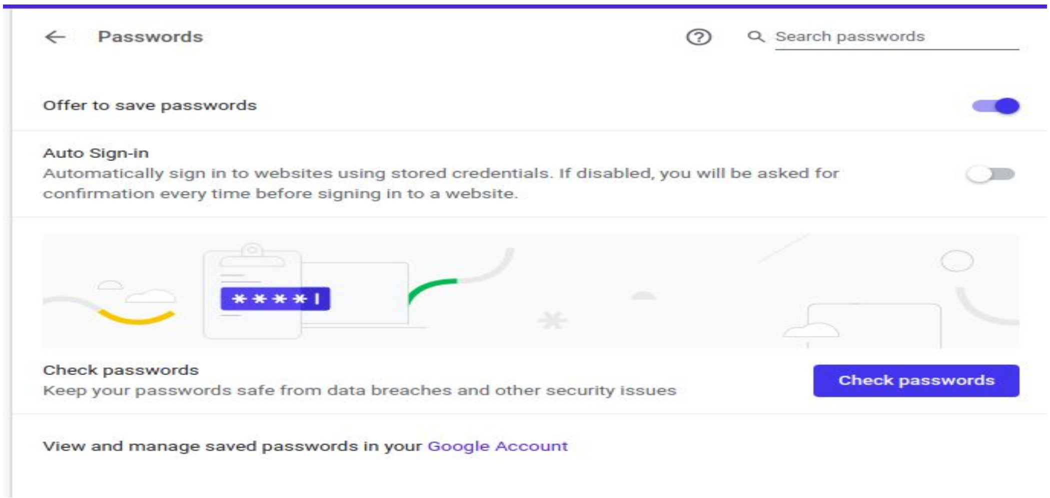
☑ Cached images and files
51.4 MB

☐ Passwords and other sign-in data
54 passwords (for tal.net, linkedin.com and 52 more, synced)

☑ Auto-fill form data

vi) Saving passwords/master password

Open Google chrome, Go to **Settings > Auto-fill> Passwords**, here we can find the list of saved passwords for different websites and those websites for which passwords are never saved. Also, we can turn on the option whether we want to be offered to save password for a new login and other related settings.



Passwords

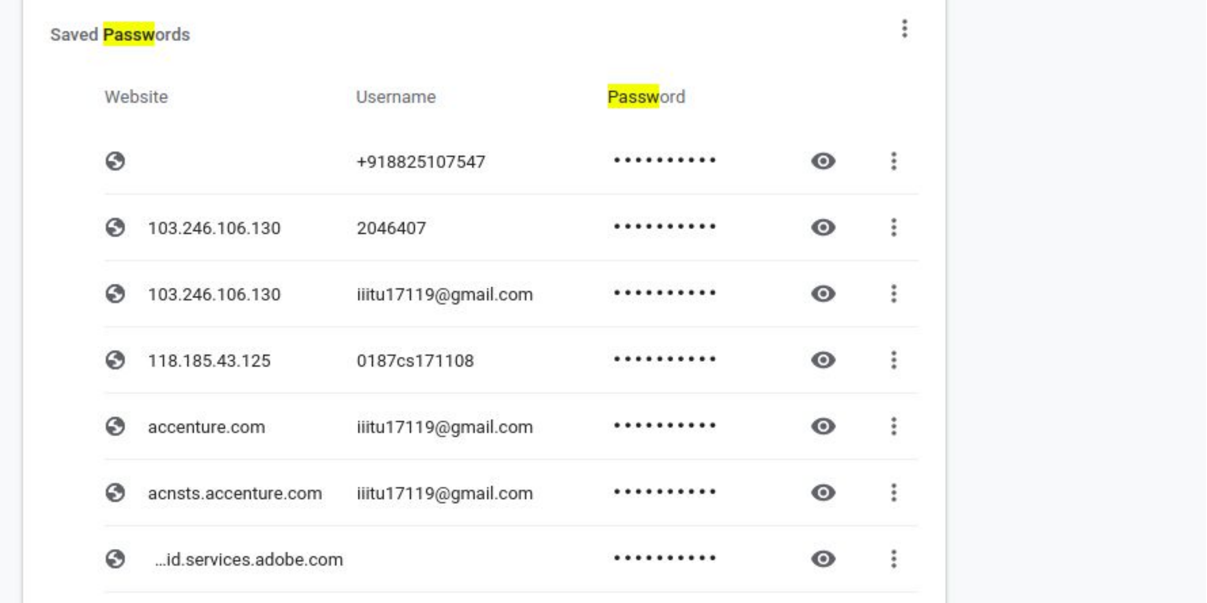
Offer to save passwords: on

Auto Sign-in: off
Automatically sign in to websites using stored credentials. If disabled, you will be asked for confirmation every time before signing in to a website.

Check passwords
Keep your passwords safe from data breaches and other security issues

View and manage saved passwords in your Google Account

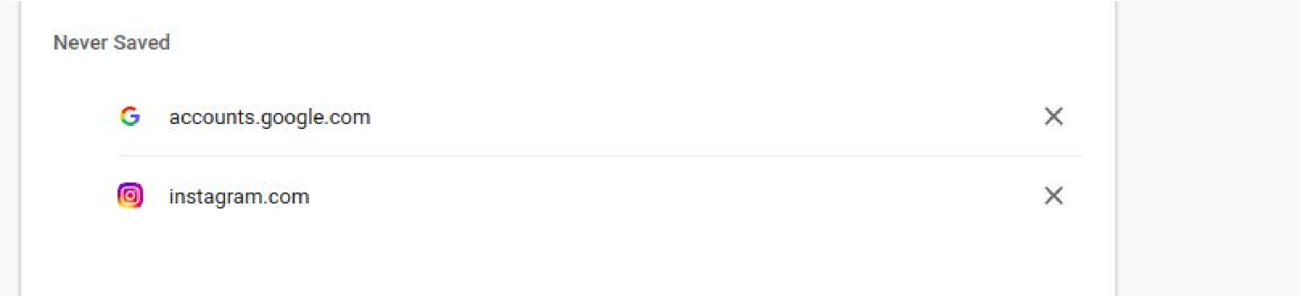
List of saved passwords.



Saved Passwords

Website	Username	Password
	+918825107547	•••••••••••
103.246.106.130	2046407	•••••••••••
103.246.106.130	iiitu17119@gmail.com	•••••••••••
118.185.43.125	0187cs171108	•••••••••••
accenture.com	iiitu17119@gmail.com	•••••••••••
acnsts.accenture.com	iiitu17119@gmail.com	•••••••••••
...id.services.adobe.com		•••••••••••

List of Websites for which password is never saved:



Never Saved

- accounts.google.com
- instagram.com

PRACTICAL 2

Perform the following tasks in Wireshark:

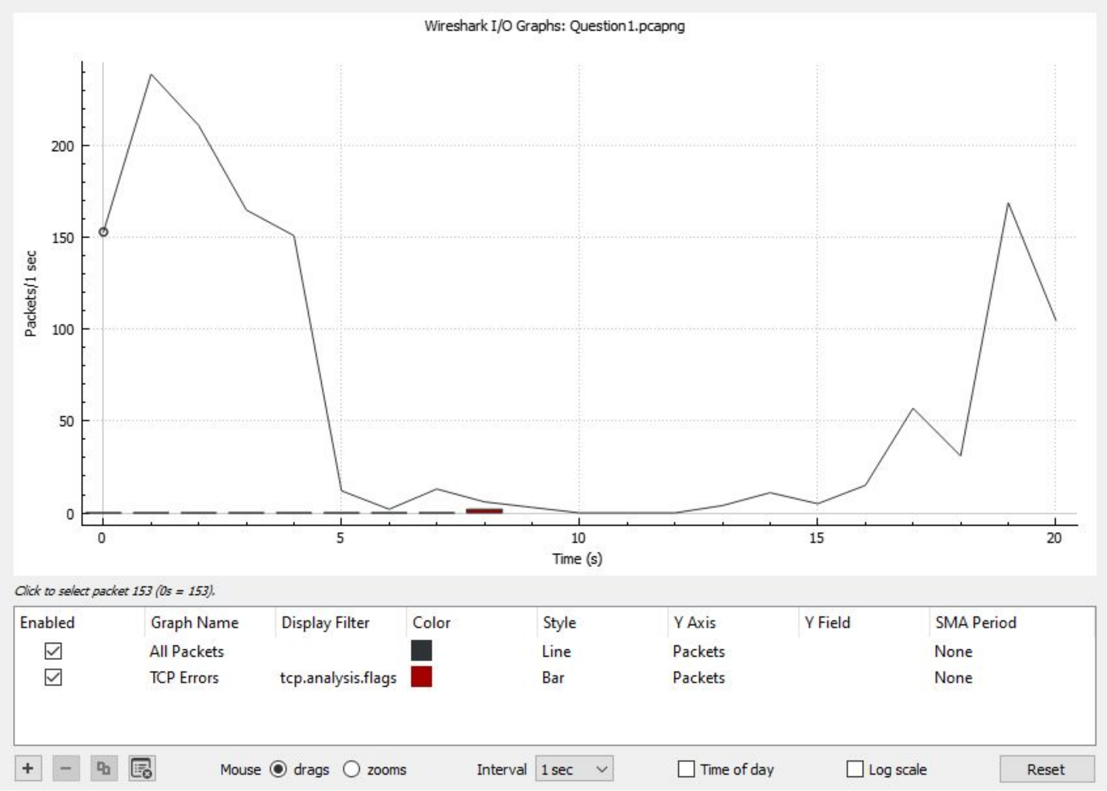
1) Capture live traffic and generate pcap file.

- Open Capture.
- Click on Start.
- Click on Stop.
- Save as question-1.pcap

  generated pcap file(Question-1).

2) Analyze the traffic using I/O graphs.

- Open Statistics
- Click on I/O graph
- Save file



Click to select packet 153 (0s = 153).

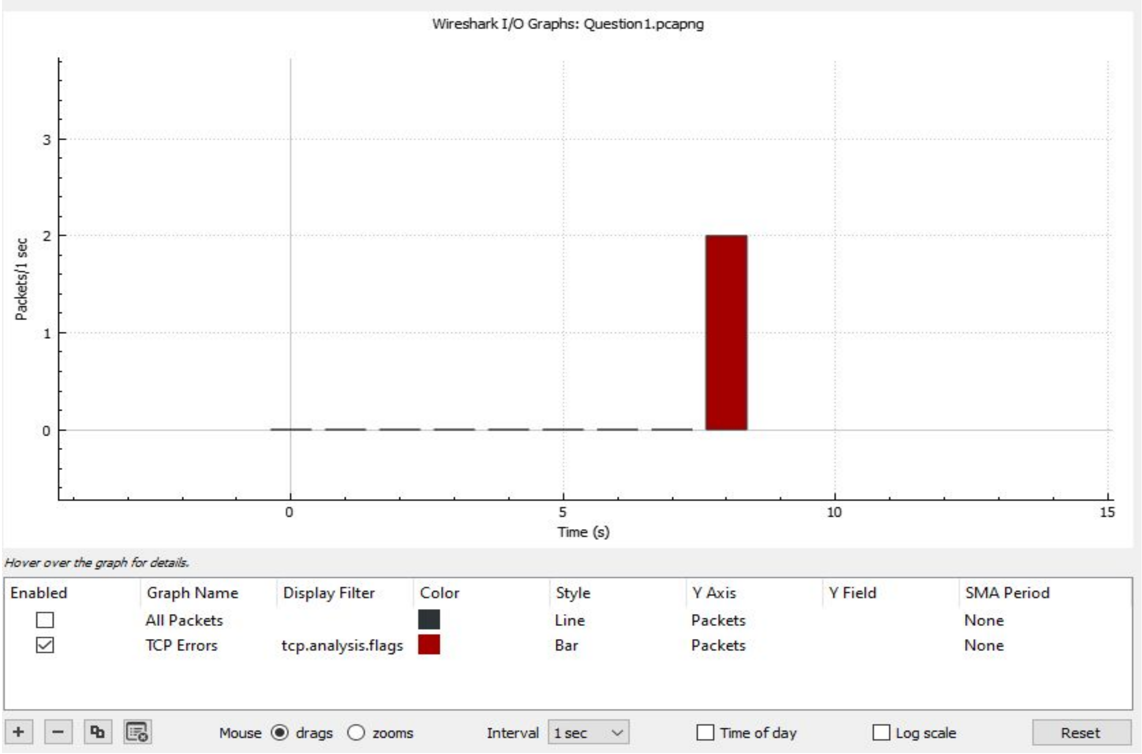
Enabled	Graph Name	Display Filter	Color	Style	Y Axis	Y Field	SMA Period
☑	All Packets		⬛	Line	Packets		None
☑	TCP Errors	tcp.analysis.flags	🟥	Bar	Packets		None

Mouse ● drags ○ zooms Interval 1 sec □ Time of day □ Log scale Reset

3) Plot the errors occurred in the traffic.

- Open Statistics.

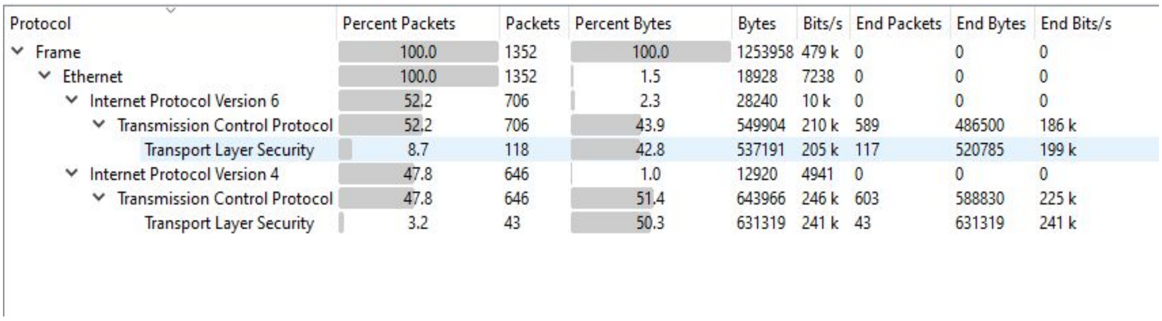
- Click on I/O graph.
- Uncheck “All Packets”
- Select or Check “TCP Errors”
- Save the file


Wireshark I/O Graphs: Question1.pcapng
Packets/1 sec
0
1
2
3
Time (s)
0
5
10
15
Hover over the graph for details.

Enabled	Graph Name	Display Filter	Color	Style	Y Axis	Y Field	SMA Period
☐	All Packets		■	Line	Packets		None
☑	TCP Errors	tcp.analysis.flags	🟥	Bar	Packets		None

4) **List the transport layer protocols in the traffic and which protocol dominates the captured traffic?**

TCP dominates the captured traffic.

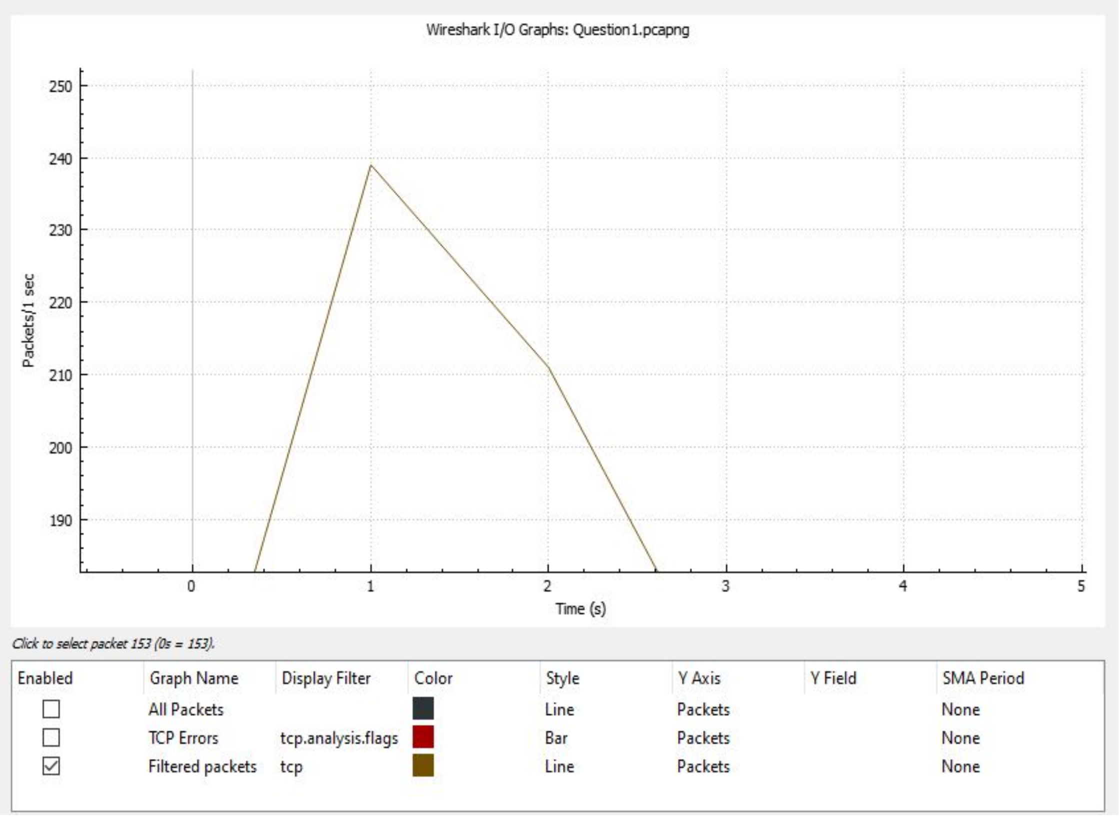


Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s
Frame	100.0	1352	100.0	1253958	479 k	0	0	0
Ethernet	100.0	1352	1.5	18928	7238	0	0	0
Internet Protocol Version 6	52.2	706	2.3	28240	10 k	0	0	0
Transmission Control Protocol	52.2	706	43.9	549904	210 k	589	486500	186 k
Transport Layer Security	8.7	118	42.8	537191	205 k	117	520785	199 k
Internet Protocol Version 4	47.8	646	1.0	12920	4941	0	0	0
Transmission Control Protocol	47.8	646	51.4	643966	246 k	603	588830	225 k
Transport Layer Security	3.2	43	50.3	631319	241 k	43	631319	241 k

5) **What is the highest number of TCP packets/sec observed? What is the peak time (in seconds)?**

- Apply filter to observe TCP packets
- Open Statistics
- Click on I/O graph

- Select Filtered packets and observe

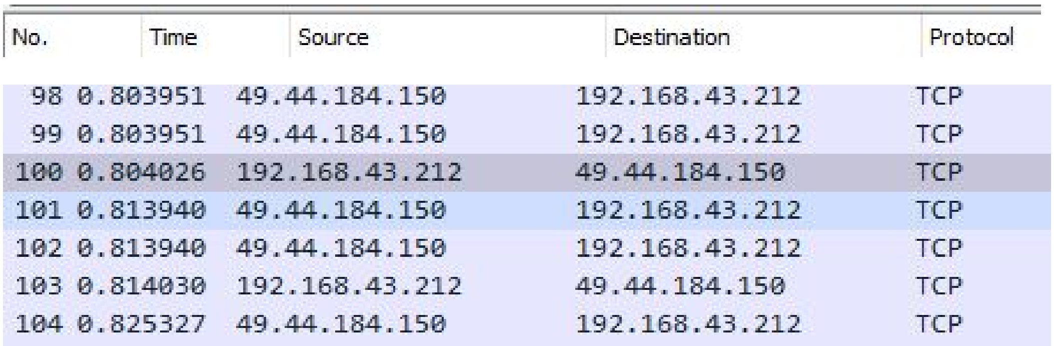

Wireshark I/O Graphs: Question1.pcapng
Packets/1 sec
250
240
230
220
210
200
190
0
1
2
3
4
5
Time (s)
Click to select packet 153 (0s = 153).

Enabled	Graph Name	Display Filter	Color	Style	Y Axis	Y Field	SMA Period
☐	All Packets		⬛	Line	Packets		None
☐	TCP Errors	tcp.analysis.flags	🟥	Bar	Packets		None
☑	Filtered packets	tcp	🟫	Line	Packets		None

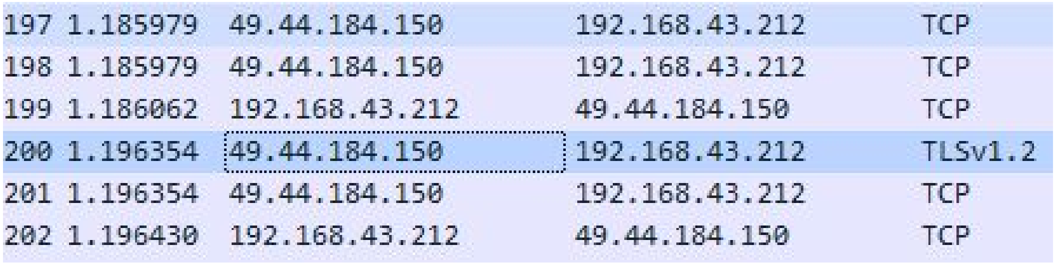
- We can see in the graph that the TCP packets/sec is 239packets/sec.
- Peak time is 1 sec.

6) Which protocol is in packet #100? What is the elapsed time from packet #100 to packet #200? How much bytes have been used during this period?

(i.) Protocol in Packet #100 is TCP.



No.	Time	Source	Destination	Protocol
98	0.803951	49.44.184.150	192.168.43.212	TCP
99	0.803951	49.44.184.150	192.168.43.212	TCP
100	0.804026	192.168.43.212	49.44.184.150	TCP
101	0.813940	49.44.184.150	192.168.43.212	TCP
102	0.813940	49.44.184.150	192.168.43.212	TCP
103	0.814030	192.168.43.212	49.44.184.150	TCP
104	0.825327	49.44.184.150	192.168.43.212	TCP

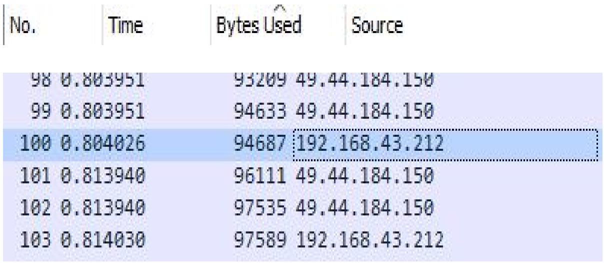


```
197 1.185979  49.44.184.150     192.168.43.212    TCP
198 1.185979  49.44.184.150     192.168.43.212    TCP
199 1.186062  192.168.43.212    49.44.184.150     TCP
200 1.196354  49.44.184.150     192.168.43.212    TLSv1.2
201 1.196354  49.44.184.150     192.168.43.212    TCP
202 1.196430  192.168.43.212    49.44.184.150     TCP
```

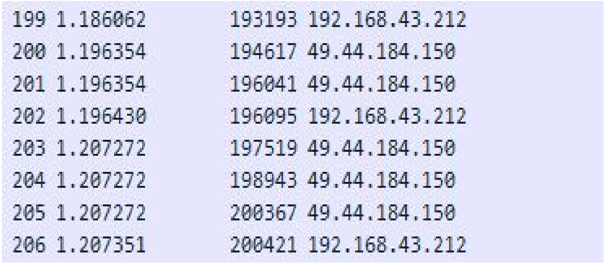
(ii.)Time elapsed from packet #100 to packet #200 is:

T_{200} - T_{100} = 1.196354 - 0.804026 = 1.159514

(iii.)Bytes Used in this period :



```
No.    Time        Bytes Used  Source
 98 0.803951       93209 49.44.184.150
 99 0.803951       94633 49.44.184.150
100 0.804026       94687 192.168.43.212
101 0.813940       96111 49.44.184.150
102 0.813940       97535 49.44.184.150
103 0.814030       97589 192.168.43.212
```



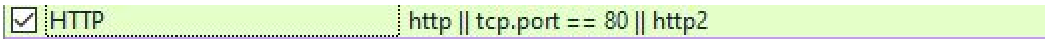
```
199 1.186062       193193 192.168.43.212
200 1.196354       194617 49.44.184.150
201 1.196354       196041 49.44.184.150
202 1.196430       196095 192.168.43.212
203 1.207272       197519 49.44.184.150
204 1.207272       198943 49.44.184.150
205 1.207272       200367 49.44.184.150
206 1.207351       200421 192.168.43.212
```

Bytes Used in this period = Bytes used in #200 - Bytes used in #100

= 194617 - 94687 = 99930 Bytes

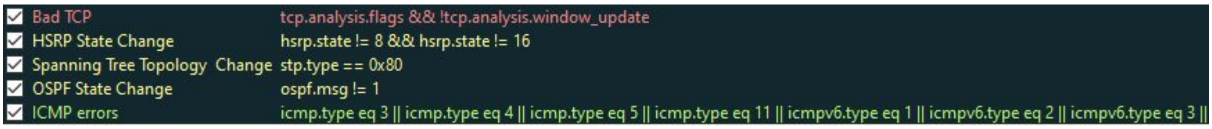
7) List the meaning of the following:

a) Packet is highlighted in green : HTTP



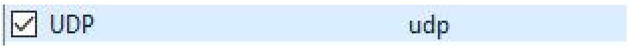
```
☑ HTTP          http || tcp.port == 80 || http2
```

b) Packet is highlighted in dark blue:



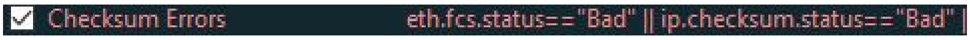
```
☑ Bad TCP                      tcp.analysis.flags && !tcp.analysis.window_update
☑ HSRP State Change            hsrp.state != 8 && hsrp.state != 16
☑ Spanning Tree Topology  Change stp.type == 0x80
☑ OSPF State Change            ospf.msg != 1
☑ ICMP errors                  icmp.type eq 3 || icmp.type eq 4 || icmp.type eq 5 || icmp.type eq 11 || icmpv6.type eq 1 || icmpv6.type eq 2 || icmpv6.type eq 3 ||
```

c) Packet is highlighted in light blue



```
☑ UDP          udp
```

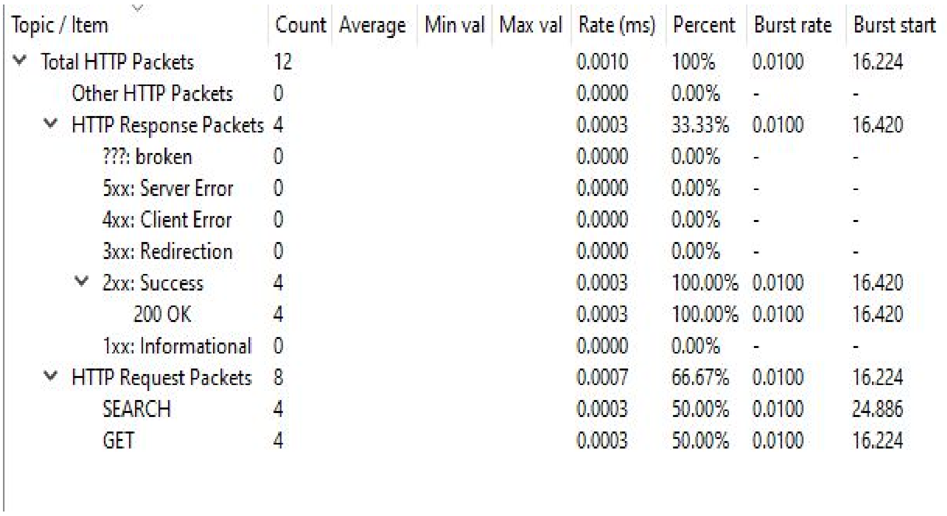
d) Packet is highlighted in black



☑ Checksum Errors eth.fcs.status=="Bad" || ip.checksum.status=="Bad" |

8) **Count the number of packets in HTTP.**

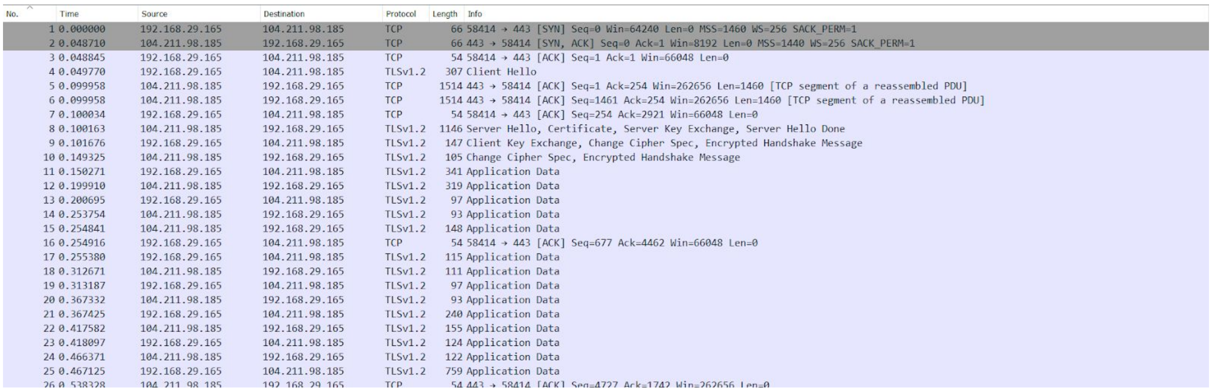
Total HTTP packets = 12



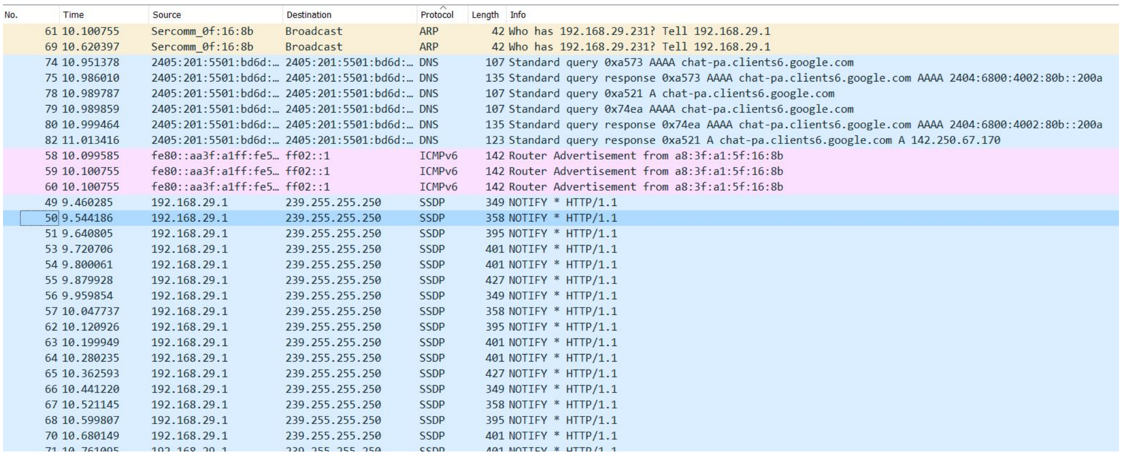
Topic / Item	Count	Average	Min val	Max val	Rate (ms)	Percent	Burst rate	Burst start
Total HTTP Packets	12				0.0010	100%	0.0100	16.224
Other HTTP Packets	0				0.0000	0.00%	-	-
HTTP Response Packets	4				0.0003	33.33%	0.0100	16.420
???: broken	0				0.0000	0.00%	-	-
5xx: Server Error	0				0.0000	0.00%	-	-
4xx: Client Error	0				0.0000	0.00%	-	-
3xx: Redirection	0				0.0000	0.00%	-	-
2xx: Success	4				0.0003	100.00%	0.0100	16.420
200 OK	4				0.0003	100.00%	0.0100	16.420
1xx: Informational	0				0.0000	0.00%	-	-
HTTP Request Packets	8				0.0007	66.67%	0.0100	16.224
SEARCH	4				0.0003	50.00%	0.0100	24.886
GET	4				0.0003	50.00%	0.0100	16.224

9) **Sort the packets by Instance ID, IP, object type, and service.**

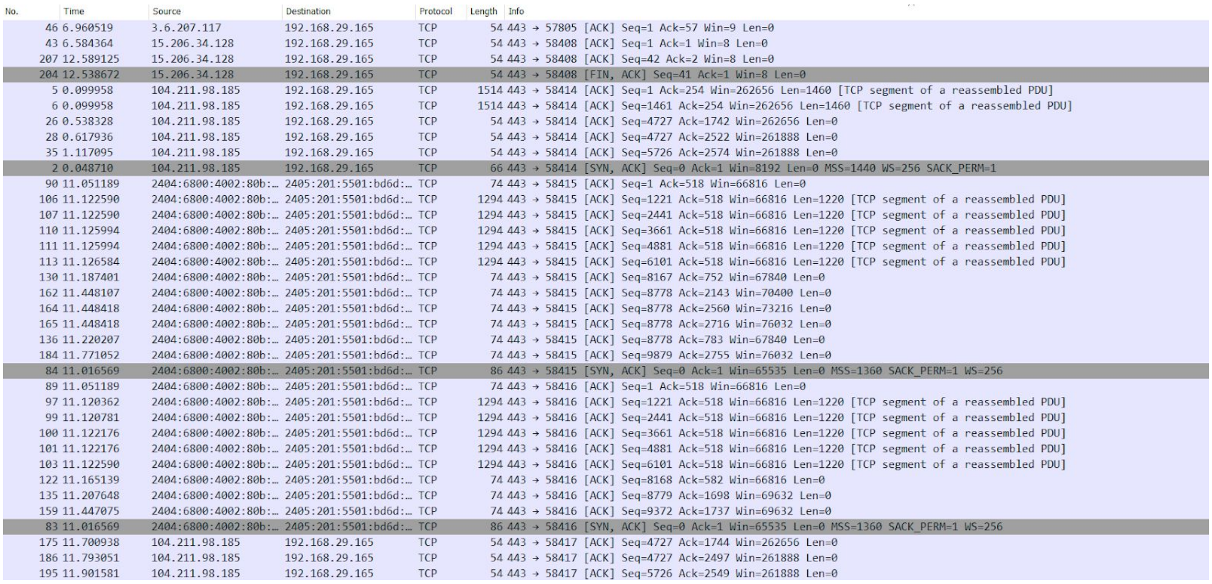
1. By Instance ID



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.29.165	104.211.98.185	TCP	66	58414 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
2	0.048710	104.211.98.185	192.168.29.165	TCP	66	443 → 58414 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1440 WS=256 SACK_PERM=1
3	0.048845	192.168.29.165	104.211.98.185	TCP	54	58414 → 443 [ACK] Seq=1 Ack=1 Win=66048 Len=0
4	0.049770	192.168.29.165	104.211.98.185	TLSv1.2	307	Client Hello
5	0.099958	104.211.98.185	192.168.29.165	TCP	1514	443 → 58414 [ACK] Seq=1 Ack=254 Win=262656 Len=1460 [TCP segment of a reassembled PDU]
6	0.099958	104.211.98.185	192.168.29.165	TCP	1514	443 → 58414 [ACK] Seq=1461 Ack=254 Win=262656 Len=1460 [TCP segment of a reassembled PDU]
7	0.100034	192.168.29.165	104.211.98.185	TCP	54	58414 → 443 [ACK] Seq=254 Ack=2921 Win=66048 Len=0
8	0.100163	104.211.98.185	192.168.29.165	TLSv1.2	1146	Server Hello, Certificate, Server Key Exchange, Server Hello Done
9	0.101676	192.168.29.165	104.211.98.185	TLSv1.2	147	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
10	0.149325	104.211.98.185	192.168.29.165	TLSv1.2	105	Change Cipher Spec, Encrypted Handshake Message
11	0.150271	192.168.29.165	104.211.98.185	TLSv1.2	341	Application Data
12	0.199910	104.211.98.185	192.168.29.165	TLSv1.2	319	Application Data
13	0.200695	192.168.29.165	104.211.98.185	TLSv1.2	97	Application Data
14	0.253754	104.211.98.185	192.168.29.165	TLSv1.2	93	Application Data
15	0.254841	104.211.98.185	192.168.29.165	TLSv1.2	148	Application Data
16	0.254916	192.168.29.165	104.211.98.185	TCP	54	58414 → 443 [ACK] Seq=677 Ack=4462 Win=66048 Len=0
17	0.255380	192.168.29.165	104.211.98.185	TLSv1.2	115	Application Data
18	0.312671	104.211.98.185	192.168.29.165	TLSv1.2	111	Application Data
19	0.313187	192.168.29.165	104.211.98.185	TLSv1.2	97	Application Data
20	0.367332	104.211.98.185	192.168.29.165	TLSv1.2	93	Application Data
21	0.367425	192.168.29.165	104.211.98.185	TLSv1.2	240	Application Data
22	0.417582	104.211.98.185	192.168.29.165	TLSv1.2	155	Application Data
23	0.418097	192.168.29.165	104.211.98.185	TLSv1.2	124	Application Data
24	0.466371	104.211.98.185	192.168.29.165	TLSv1.2	122	Application Data
25	0.467125	192.168.29.165	104.211.98.185	TLSv1.2	759	Application Data
26	0.538328	104.211.98.185	192.168.29.165	TCP	54	443 → 58414 [ACK] Seq=4727 Ack=1742 Win=262656 Len=0



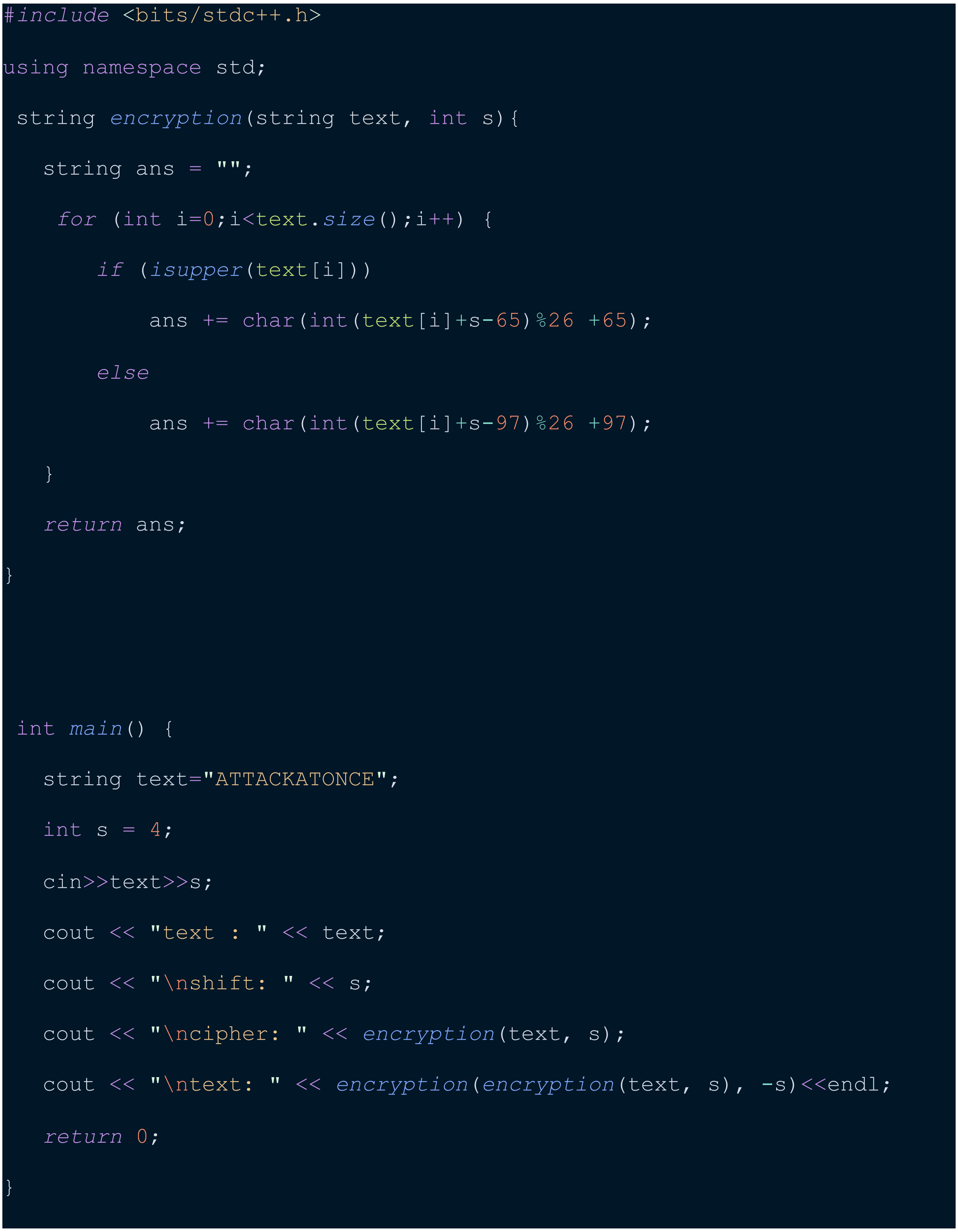
```
No.     Time       Source                 Destination            Protocol  Length Info
     61 10.100755  Sercomm_0f:16:8b       Broadcast              ARP           42 Who has 192.168.29.231? Tell 192.168.29.1
     69 10.620397  Sercomm_0f:16:8b       Broadcast              ARP           42 Who has 192.168.29.231? Tell 192.168.29.1
     74 10.951378  2405:201:5501:bd6d:…   2405:201:5501:bd6d:…   DNS          107 Standard query 0xa573 AAAA chat-pa.clients6.google.com
     75 10.986010  2405:201:5501:bd6d:…   2405:201:5501:bd6d:…   DNS          135 Standard query response 0xa573 AAAA chat-pa.clients6.google.com AAAA 2404:6800:4002:80b::200a
     78 10.989787  2405:201:5501:bd6d:…   2405:201:5501:bd6d:…   DNS          107 Standard query 0xa521 A chat-pa.clients6.google.com
     79 10.989859  2405:201:5501:bd6d:…   2405:201:5501:bd6d:…   DNS          107 Standard query 0x74ea AAAA chat-pa.clients6.google.com
     80 10.999464  2405:201:5501:bd6d:…   2405:201:5501:bd6d:…   DNS          135 Standard query response 0x74ea AAAA chat-pa.clients6.google.com AAAA 2404:6800:4002:80b::200a
     82 11.013416  2405:201:5501:bd6d:…   2405:201:5501:bd6d:…   DNS          123 Standard query response 0xa521 A chat-pa.clients6.google.com A 142.250.67.170
     58 10.099585  fe80::aa3f:a1ff:fe5…   ff02::1                ICMPv6       142 Router Advertisement from a8:3f:a1:5f:16:8b
     59 10.100755  fe80::aa3f:a1ff:fe5…   ff02::1                ICMPv6       142 Router Advertisement from a8:3f:a1:5f:16:8b
     60 10.100755  fe80::aa3f:a1ff:fe5…   ff02::1                ICMPv6       142 Router Advertisement from a8:3f:a1:5f:16:8b
     49 9.460285   192.168.29.1           239.255.255.250        SSDP         349 NOTIFY * HTTP/1.1
     50 9.544186   192.168.29.1           239.255.255.250        SSDP         358 NOTIFY * HTTP/1.1
     51 9.640805   192.168.29.1           239.255.255.250        SSDP         395 NOTIFY * HTTP/1.1
     53 9.720706   192.168.29.1           239.255.255.250        SSDP         401 NOTIFY * HTTP/1.1
     54 9.800061   192.168.29.1           239.255.255.250        SSDP         401 NOTIFY * HTTP/1.1
     55 9.879928   192.168.29.1           239.255.255.250        SSDP         427 NOTIFY * HTTP/1.1
     56 9.959854   192.168.29.1           239.255.255.250        SSDP         349 NOTIFY * HTTP/1.1
     57 10.047737  192.168.29.1           239.255.255.250        SSDP         358 NOTIFY * HTTP/1.1
     62 10.120926  192.168.29.1           239.255.255.250        SSDP         395 NOTIFY * HTTP/1.1
     63 10.199949  192.168.29.1           239.255.255.250        SSDP         401 NOTIFY * HTTP/1.1
     64 10.280235  192.168.29.1           239.255.255.250        SSDP         401 NOTIFY * HTTP/1.1
     65 10.362593  192.168.29.1           239.255.255.250        SSDP         427 NOTIFY * HTTP/1.1
     66 10.441220  192.168.29.1           239.255.255.250        SSDP         349 NOTIFY * HTTP/1.1
     67 10.521145  192.168.29.1           239.255.255.250        SSDP         358 NOTIFY * HTTP/1.1
     68 10.599807  192.168.29.1           239.255.255.250        SSDP         395 NOTIFY * HTTP/1.1
     70 10.680149  192.168.29.1           239.255.255.250        SSDP         401 NOTIFY * HTTP/1.1
     71 10.761095  192.168.29.1           239.255.255.250        SSDP         401 NOTIFY * HTTP/1.1
```



```
No.     Time       Source                 Destination            Protocol  Length Info
     46 6.960519   3.6.207.117            192.168.29.165         TCP           54 443 → 57805 [ACK] Seq=1 Ack=57 Win=9 Len=0
     43 6.584364   15.206.34.128          192.168.29.165         TCP           54 443 → 58408 [ACK] Seq=1 Ack=1 Win=8 Len=0
    207 12.589125  15.206.34.128          192.168.29.165         TCP           54 443 → 58408 [ACK] Seq=42 Ack=2 Win=8 Len=0
    204 12.538672  15.206.34.128          192.168.29.165         TCP           54 443 → 58408 [FIN, ACK] Seq=41 Ack=1 Win=8 Len=0
      5 0.099958   104.211.98.185         192.168.29.165         TCP         1514 443 → 58414 [ACK] Seq=1 Ack=254 Win=262656 Len=1460 [TCP segment of a reassembled PDU]
      6 0.099958   104.211.98.185         192.168.29.165         TCP         1514 443 → 58414 [ACK] Seq=1461 Ack=254 Win=262656 Len=1460 [TCP segment of a reassembled PDU]
     26 0.538328   104.211.98.185         192.168.29.165         TCP           54 443 → 58414 [ACK] Seq=4727 Ack=1742 Win=262656 Len=0
     28 0.617936   104.211.98.185         192.168.29.165         TCP           54 443 → 58414 [ACK] Seq=4727 Ack=2522 Win=261888 Len=0
     35 1.117095   104.211.98.185         192.168.29.165         TCP           54 443 → 58414 [ACK] Seq=5726 Ack=2574 Win=261888 Len=0
      2 0.048710   104.211.98.185         192.168.29.165         TCP           66 443 → 58414 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1440 WS=256 SACK_PERM=1
     90 11.051189  2404:6800:4002:80b:…   2405:201:5501:bd6d:…   TCP           74 443 → 58415 [ACK] Seq=1 Ack=518 Win=66816 Len=0
    106 11.122590  2404:6800:4002:80b:…   2405:201:5501:bd6d:…   TCP         1294 443 → 58415 [ACK] Seq=1221 Ack=518 Win=66816 Len=1220 [TCP segment of a reassembled PDU]
    107 11.122590  2404:6800:4002:80b:…   2405:201:5501:bd6d:…   TCP         1294 443 → 58415 [ACK] Seq=2441 Ack=518 Win=66816 Len=1220 [TCP segment of a reassembled PDU]
    110 11.125994  2404:6800:4002:80b:…   2405:201:5501:bd6d:…   TCP         1294 443 → 58415 [ACK] Seq=3661 Ack=518 Win=66816 Len=1220 [TCP segment of a reassembled PDU]
    111 11.125994  2404:6800:4002:80b:…   2405:201:5501:bd6d:…   TCP         1294 443 → 58415 [ACK] Seq=4881 Ack=518 Win=66816 Len=1220 [TCP segment of a reassembled PDU]
    113 11.126584  2404:6800:4002:80b:…   2405:201:5501:bd6d:…   TCP         1294 443 → 58415 [ACK] Seq=6101 Ack=518 Win=66816 Len=1220 [TCP segment of a reassembled PDU]
    130 11.187401  2404:6800:4002:80b:…   2405:201:5501:bd6d:…   TCP           74 443 → 58415 [ACK] Seq=8167 Ack=752 Win=67840 Len=0
    162 11.448107  2404:6800:4002:80b:…   2405:201:5501:bd6d:…   TCP           74 443 → 58415 [ACK] Seq=8778 Ack=2143 Win=70400 Len=0
    164 11.448418  2404:6800:4002:80b:…   2405:201:5501:bd6d:…   TCP           74 443 → 58415 [ACK] Seq=8778 Ack=2560 Win=73216 Len=0
    165 11.448418  2404:6800:4002:80b:…   2405:201:5501:bd6d:…   TCP           74 443 → 58415 [ACK] Seq=8778 Ack=2716 Win=76032 Len=0
    136 11.220207  2404:6800:4002:80b:…   2405:201:5501:bd6d:…   TCP           74 443 → 58415 [ACK] Seq=8778 Ack=783 Win=67840 Len=0
    184 11.771052  2404:6800:4002:80b:…   2405:201:5501:bd6d:…   TCP           74 443 → 58415 [ACK] Seq=9879 Ack=2755 Win=76032 Len=0
     84 11.016569  2404:6800:4002:80b:…   2405:201:5501:bd6d:…   TCP           86 443 → 58415 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1360 SACK_PERM=1 WS=256
     89 11.051189  2404:6800:4002:80b:…   2405:201:5501:bd6d:…   TCP           74 443 → 58416 [ACK] Seq=1 Ack=518 Win=66816 Len=0
     97 11.120362  2404:6800:4002:80b:…   2405:201:5501:bd6d:…   TCP         1294 443 → 58416 [ACK] Seq=1221 Ack=518 Win=66816 Len=1220 [TCP segment of a reassembled PDU]
     99 11.120781  2404:6800:4002:80b:…   2405:201:5501:bd6d:…   TCP         1294 443 → 58416 [ACK] Seq=2441 Ack=518 Win=66816 Len=1220 [TCP segment of a reassembled PDU]
    100 11.122176  2404:6800:4002:80b:…   2405:201:5501:bd6d:…   TCP         1294 443 → 58416 [ACK] Seq=3661 Ack=518 Win=66816 Len=1220 [TCP segment of a reassembled PDU]
    101 11.122176  2404:6800:4002:80b:…   2405:201:5501:bd6d:…   TCP         1294 443 → 58416 [ACK] Seq=4881 Ack=518 Win=66816 Len=1220 [TCP segment of a reassembled PDU]
    103 11.122590  2404:6800:4002:80b:…   2405:201:5501:bd6d:…   TCP         1294 443 → 58416 [ACK] Seq=6101 Ack=518 Win=66816 Len=1220 [TCP segment of a reassembled PDU]
    122 11.165139  2404:6800:4002:80b:…   2405:201:5501:bd6d:…   TCP           74 443 → 58416 [ACK] Seq=8168 Ack=582 Win=66816 Len=0
    135 11.207648  2404:6800:4002:80b:…   2405:201:5501:bd6d:…   TCP           74 443 → 58416 [ACK] Seq=8779 Ack=1698 Win=69632 Len=0
    159 11.447075  2404:6800:4002:80b:…   2405:201:5501:bd6d:…   TCP           74 443 → 58416 [ACK] Seq=9372 Ack=1737 Win=69632 Len=0
     83 11.016569  2404:6800:4002:80b:…   2405:201:5501:bd6d:…   TCP           86 443 → 58416 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1360 SACK_PERM=1 WS=256
    175 11.700938  104.211.98.185         192.168.29.165         TCP           54 443 → 58417 [ACK] Seq=4727 Ack=1744 Win=262656 Len=0
    186 11.793051  104.211.98.185         192.168.29.165         TCP           54 443 → 58417 [ACK] Seq=4727 Ack=2497 Win=261888 Len=0
    195 11.901581  104.211.98.185         192.168.29.165         TCP           54 443 → 58417 [ACK] Seq=5726 Ack=2549 Win=261888 Len=0
```

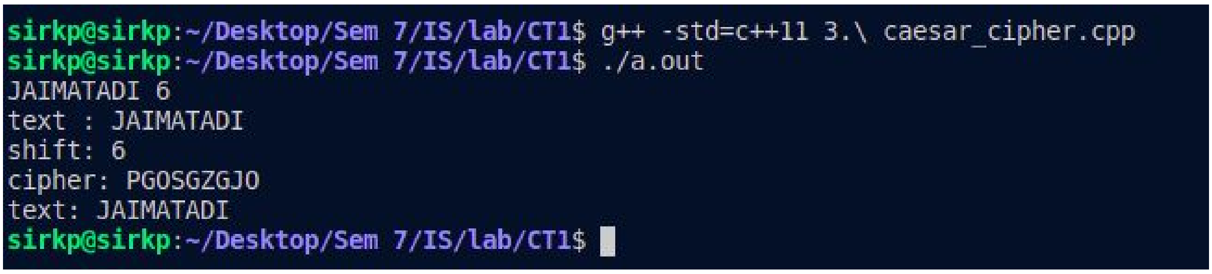
PRACTICAL 3

Write a program for implementation of Caesar cipher cryptosystem.



```cpp
#include <bits/stdc++.h>

using namespace std;

 string encryption(string text, int s){

   string ans = "";

    for (int i=0;i<text.size();i++) {

       if (isupper(text[i]))

           ans += char(int(text[i]+s-65)%26 +65);

       else

           ans += char(int(text[i]+s-97)%26 +97);

   }

   return ans;

}



 int main() {

   string text="ATTACKATONCE";

   int s = 4;

   cin>>text>>s;

   cout << "text : " << text;

   cout << "\nshift: " << s;

   cout << "\ncipher: " << encryption(text, s);

   cout << "\ntext: " << encryption(encryption(text, s), -s)<<endl;

   return 0;

}
```

OUTPUT



```
sirkp@sirkp:~/Desktop/Sem 7/IS/lab/CT1$ g++ -std=c++11 3.\ caesar_cipher.cpp
sirkp@sirkp:~/Desktop/Sem 7/IS/lab/CT1$ ./a.out
JAIMATADI 6
text : JAIMATADI
shift: 6
cipher: PGOSGZGJO
text: JAIMATADI
sirkp@sirkp:~/Desktop/Sem 7/IS/lab/CT1$
```

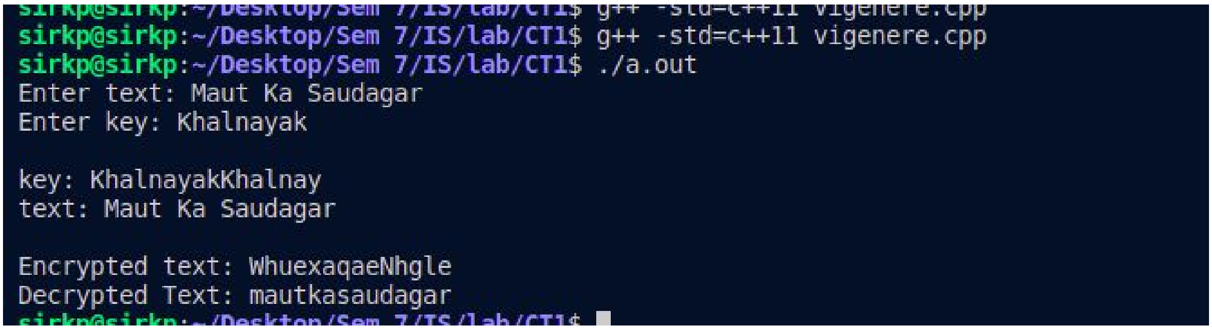
PRACTICAL 4

Write a program for implementation of Vigenère cryptosystem.

```
#include<bits/stdc++.h>
using namespace std;

void process(string& text){
   string temp;
   for(auto c: text){
       if(c == 32)
           continue;
       temp.push_back(c);
   }
   transform(temp.begin(),temp.end(),temp.begin(),::tolower);
   text = temp;
}

string keyStream(string key, string text){
   int key_len = key.size();
   string new_key(key);

   int i=0;
   while(new_key.size()!=text.size()){
       new_key.push_back(key[i]);
       i++;
       if(i==key_len)i=0;

   }
   return new_key;
```

```
}

string vigenere(string& text, string key){
   process(text);
   string encrypted;

   for (int i = 0;i<text.size(); i++){
       char p = (text[i] + key[i]-2*'a')%26;
       p+='a';
       encrypted.push_back(p);
   }
   return encrypted;
}

string decryption(string encrypted,string key){
   string decrypted;

   for (int i = 0;i<encrypted.size(); i++){
       char p = (encrypted[i] -key[i] +26) %26;
       p+='a';
       decrypted.push_back(p);
   }
   return decrypted;
}

int main(){
   string key, text;
   cout<<"Enter text: ";
   getline(cin, text);
   cout<<"Enter key: ";

   cin >> key;

   cout<<endl;

   key = keyStream(key,text);

   cout << "key: " << key << endl;

   cout << "text: " << text << endl;

   string encrypted = vigenere(text,key);


   cout << "\nEncrypted text: " << encrypted << endl;

   string decrypted = decryption(encrypted,key) ;


   cout << "Decrypted Text: " << decrypted << endl;

   return 0;

}
```

OUTPUT

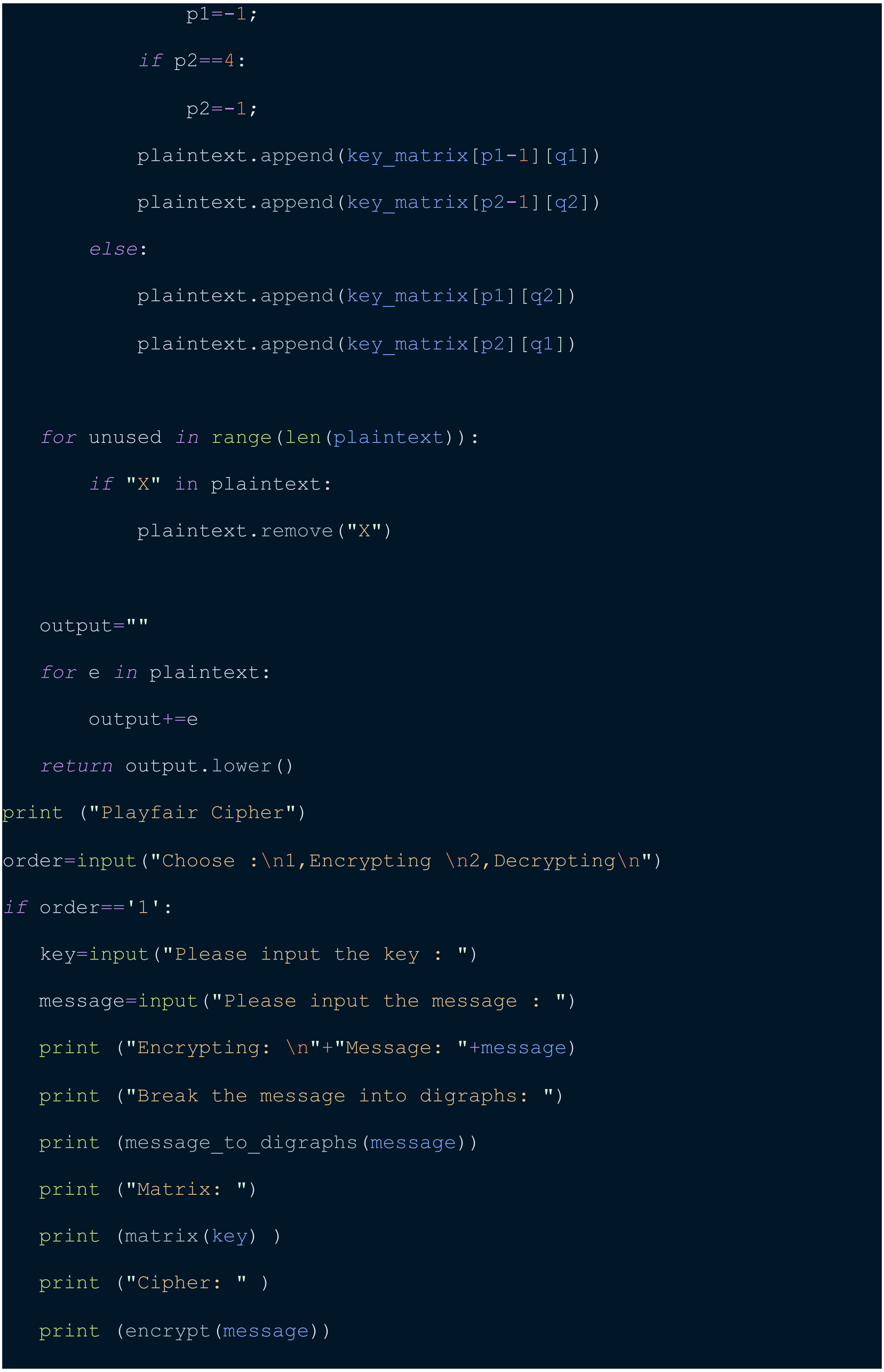


```
sirkp@sirkp:~/Desktop/Sem 7/IS/lab/CT1$ g++ -std=c++11 vigenere.cpp
sirkp@sirkp:~/Desktop/Sem 7/IS/lab/CT1$ g++ -std=c++11 vigenere.cpp
sirkp@sirkp:~/Desktop/Sem 7/IS/lab/CT1$ ./a.out
Enter text: Maut Ka Saudagar
Enter key: Khalnayak

key: KhalnayakKhalnay
text: Maut Ka Saudagar

Encrypted text: WhuexaqaeNhgle
Decrypted Text: mautkasaudagar
sirkp@sirkp:~/Desktop/Sem 7/IS/lab/CT1$
```

PRACTICAL 5

Write a program for implementation of Playfair cryptosystem.

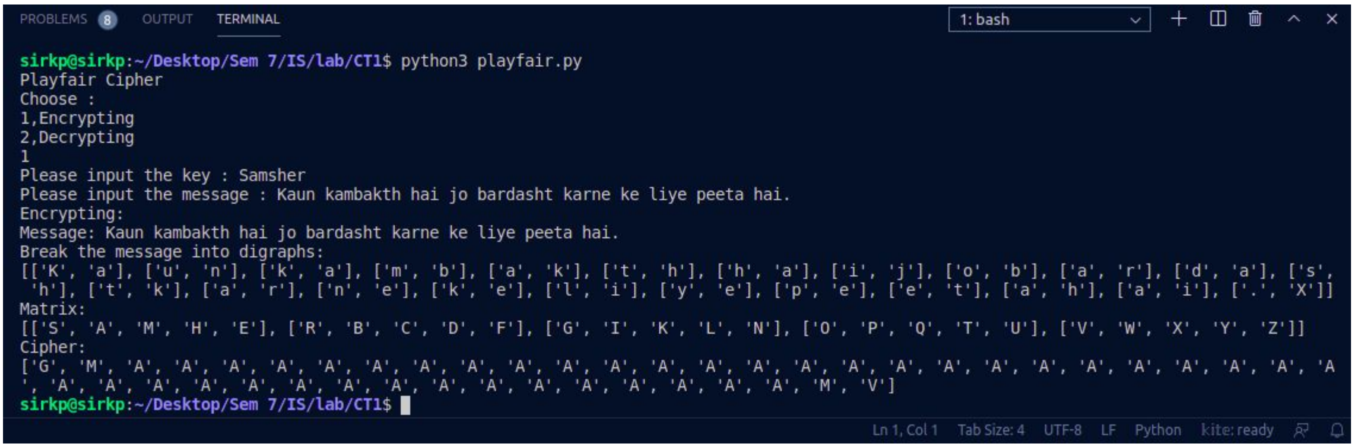
```
def matrix(key):
    matrix=[]
    for e in key.upper():
        if e not in matrix:
            matrix.append(e)
    alphabet="ABCDEFGHIKLMNOPQRSTUVWXYZ"

    for e in alphabet:
        if e not in matrix:
            matrix.append(e)

    #initialize a new list. Is there any elegant way to do that?
    matrix_group=[]
    for e in range(5):
        matrix_group.append('')

    #Break it into 5*5
    matrix_group[0]=matrix[0:5]
    matrix_group[1]=matrix[5:10]
    matrix_group[2]=matrix[10:15]
    matrix_group[3]=matrix[15:20]
    matrix_group[4]=matrix[20:25]
    return matrix_group

def message_to_digraphs(message_original):
    #Change it to Array. Because I want used insert() method
    message=[]
    for e in message_original:

        message.append(e)



    #Delet space

    for unused in range(len(message)):

        if " " in message:

            message.remove(" ")



    #If both letters are the same, add an "X" after the first letter.

    i=0

    for e in range(int(len(message)/2)):

        if message[i]==message[i+1]:

            message.insert(i+1,'X')

        i=i+2



    #If it is odd digit, add an "X" at the end

    if len(message)%2==1:

        message.append("X")

    #Grouping

    i=0

    new=[]

    for x in range(1,int(len(message)/2)+1):

        new.append(message[i:i+2])

        i=i+2

    return new



def find_position(key_matrix,letter):

    x=y=0

    for i in range(5):
        for j in range(5):
            if key_matrix[i][j]==letter:
                x=i
                y=j

    return x,y

def encrypt(message):
    message=message_to_digraphs(message)
    key_matrix=matrix(key)
    cipher=[]
    for e in message:
        p1,q1=find_position(key_matrix,e[0])
        p2,q2=find_position(key_matrix,e[1])
        if p1==p2:
            if q1==4:
                q1=-1
            if q2==4:
                q2=-1
            cipher.append(key_matrix[p1][q1+1])
            cipher.append(key_matrix[p1][q2+1])
        elif q1==q2:
            if p1==4:
                p1=-1;
            if p2==4:
                p2=-1;
            cipher.append(key_matrix[p1+1][q1])
            cipher.append(key_matrix[p2+1][q2])
        else:
            cipher.append(key_matrix[p1][q2])

            cipher.append(key_matrix[p2][q1])

    return cipher


def cipher_to_digraphs(cipher):

    i=0

    new=[]

    for x in range(len(cipher)/2):

        new.append(cipher[i:i+2])

        i=i+2

    return new




def decrypt(cipher):

    cipher=cipher_to_digraphs(cipher)

    key_matrix=matrix(key)

    plaintext=[]

    for e in cipher:

        p1,q1=find_position(key_matrix,e[0])

        p2,q2=find_position(key_matrix,e[1])

        if p1==p2:

            if q1==4:

                q1=-1

            if q2==4:

                q2=-1

            plaintext.append(key_matrix[p1][q1-1])

            plaintext.append(key_matrix[p1][q2-1])

        elif q1==q2:

            if p1==4:
```



```
                p1=-1;
            if p2==4:
                p2=-1;
            plaintext.append(key_matrix[p1-1][q1])
            plaintext.append(key_matrix[p2-1][q2])
        else:
            plaintext.append(key_matrix[p1][q2])
            plaintext.append(key_matrix[p2][q1])

    for unused in range(len(plaintext)):
        if "X" in plaintext:
            plaintext.remove("X")

    output=""
    for e in plaintext:
        output+=e
    return output.lower()
print ("Playfair Cipher")
order=input("Choose :\n1,Encrypting \n2,Decrypting\n")
if order=='1':
    key=input("Please input the key : ")
    message=input("Please input the message : ")
    print ("Encrypting: \n"+"Message: "+message)
    print ("Break the message into digraphs: ")
    print (message_to_digraphs(message))
    print ("Matrix: ")
    print (matrix(key) )
    print ("Cipher: " )
    print (encrypt(message))
```

```
elif order=='2':

   key=input("Please input the key : ")

   cipher=input("Please input the cipher text: ")

   #cipher="ILSYQFBWBMLIAFFQ"

   print ("\nDecrypting: \n"+"Cipher: "+cipher)

   print ("Plaintext:")

   print (decrypt(cipher))

else:

   print ("Error")
```

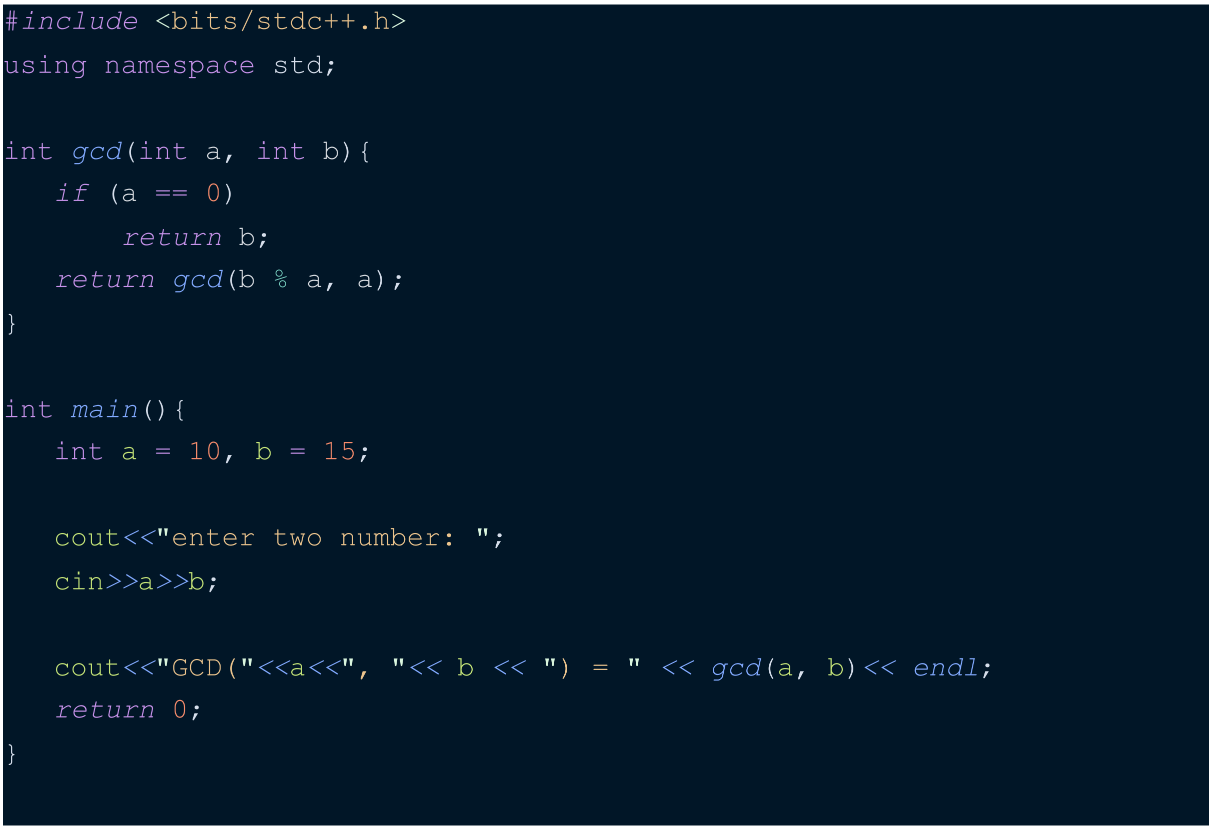
OUTPUT



```
sirkp@sirkp:~/Desktop/Sem 7/IS/lab/CT1$ python3 playfair.py
Playfair Cipher
Choose :
1,Encrypting
2,Decrypting
1
Please input the key : Samsher
Please input the message : Kaun kambakth hai jo bardasht karne ke liye peeta hai.
Encrypting:
Message: Kaun kambakth hai jo bardasht karne ke liye peeta hai.
Break the message into digraphs:
[['K', 'a'], ['u', 'n'], ['k', 'a'], ['m', 'b'], ['a', 'k'], ['t', 'h'], ['h', 'a'], ['i', 'j'], ['o', 'b'], ['a', 'r'], ['d', 'a'], ['s',
 'h'], ['t', 'k'], ['a', 'r'], ['n', 'e'], ['k', 'e'], ['l', 'i'], ['y', 'e'], ['p', 'e'], ['e', 't'], ['a', 'h'], ['a', 'i'], ['.', 'X']]
Matrix:
[['S', 'A', 'M', 'H', 'E'], ['R', 'B', 'C', 'D', 'F'], ['G', 'I', 'K', 'L', 'N'], ['O', 'P', 'Q', 'T', 'U'], ['V', 'W', 'X', 'Y', 'Z']]
Cipher:
['G', 'M', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A
', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'M', 'V']
sirkp@sirkp:~/Desktop/Sem 7/IS/lab/CT1$ 
```

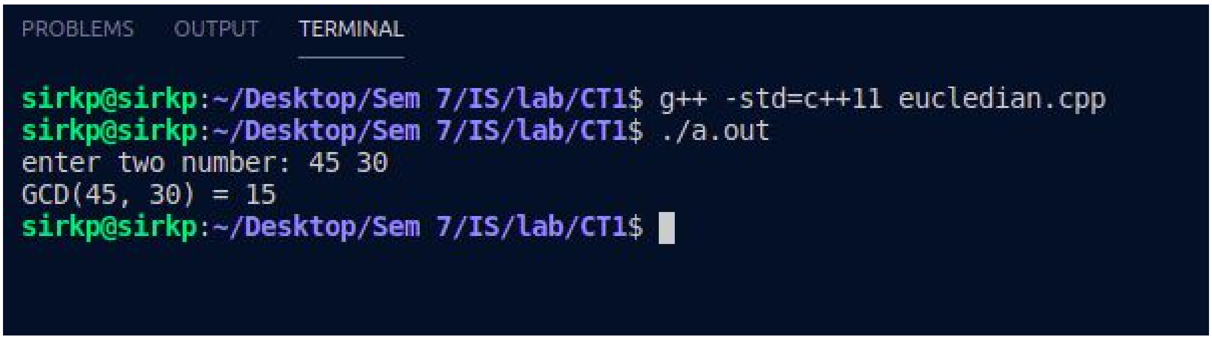
PRACTICAL 6

Write a program for implementation of Euclid’s algorithm.



```
#include <bits/stdc++.h>
using namespace std;

int gcd(int a, int b){
   if (a == 0)
       return b;
   return gcd(b % a, a);
}

int main(){
   int a = 10, b = 15;

   cout<<"enter two number: ";
   cin>>a>>b;

   cout<<"GCD("<<a<<", "<< b << ") = " << gcd(a, b)<< endl;
   return 0;
}
```

OUTPUT



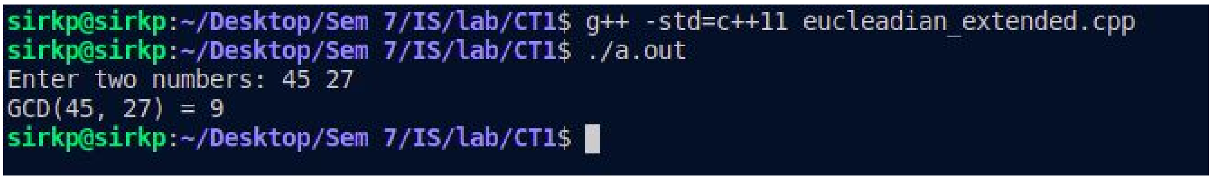
```
sirkp@sirkp:~/Desktop/Sem 7/IS/lab/CT1$ g++ -std=c++11 eucledian.cpp
sirkp@sirkp:~/Desktop/Sem 7/IS/lab/CT1$ ./a.out
enter two number: 45 30
GCD(45, 30) = 15
sirkp@sirkp:~/Desktop/Sem 7/IS/lab/CT1$ 
```

PRACTICAL 7

Write a program for implementation of Extended Euclidean algorithm.

```
#include <bits/stdc++.h>
using namespace std;

int gcdExtended(int a, int b, int& x, int& y) {
   if (a == 0) {
       x = 0;
       y = 1;
       return b;
   }

   int x1, y1;
   int gcd = gcdExtended(b%a, a, x1, y1);

   x = y1 - (b/a) * x1;
   y = x1;

   return gcd;
}

int main(){
   int x, y, a = 35, b = 15;
   cout<<"Enter two numbers: ";
   cin>>a>>b;
   int g = gcdExtended(a, b, x, y);
   cout<<"GCD("<<a<<", "<<b<< ") = "<<g<<endl;
   return 0;}
```

OUTPUT

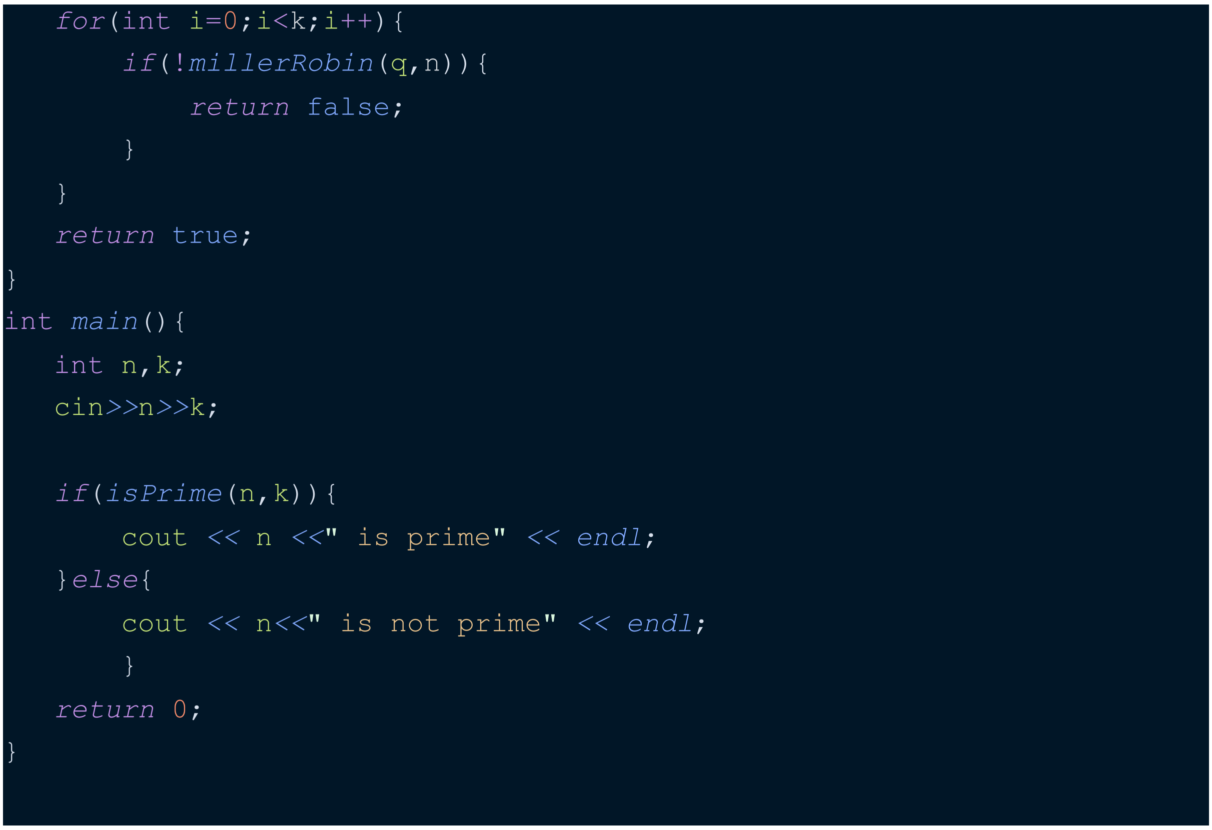


```
sirkp@sirkp:~/Desktop/Sem 7/IS/lab/CT1$ g++ -std=c++11 eucleadian_extended.cpp
sirkp@sirkp:~/Desktop/Sem 7/IS/lab/CT1$ ./a.out
Enter two numbers: 45 27
GCD(45, 27) = 9
sirkp@sirkp:~/Desktop/Sem 7/IS/lab/CT1$
```

PRACTICAL 8

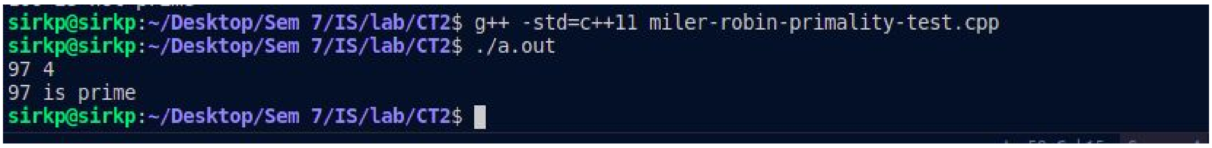
Write a program for implementation of Rabin-Miller Primality Test.

```cpp
#include<bits/stdc++.h>
using namespace std;
#define ll long long

ll power(ll a,ll b,ll mod){
    a=a%mod;
    ll ans=1;
    while(b>0){
        if(b&1){
            ans=(ans*a)%mod;
        }
        b=b>>1; a=(a*a)%mod;
    }
    return ans;
}
bool millerRobin(int d,int n){
    int a = 2+rand()%(n-4);

    int x = power(a,d,n);

    if(x==1 or x==n-1) return true;

    while(d!=n-1){
        x = (x*x)%n;
        d*=2;

        if(x==1 or x==n-1)return true;
    }
    return false;
}

bool isPrime(int n,int k){
    if(n==2 or n==3)return true;
    if(n<=4)return false;

    int q = n-1;
    while(q%2!=1){
        q/=2;
    }

```



```
    for(int i=0;i<k;i++){
        if(!millerRobin(q,n)){
            return false;
        }
    }
    return true;
}
int main(){
    int n,k;
    cin>>n>>k;

    if(isPrime(n,k)){
        cout << n <<" is prime" << endl;
    }else{
        cout << n<<" is not prime" << endl;
        }
    return 0;
}
```

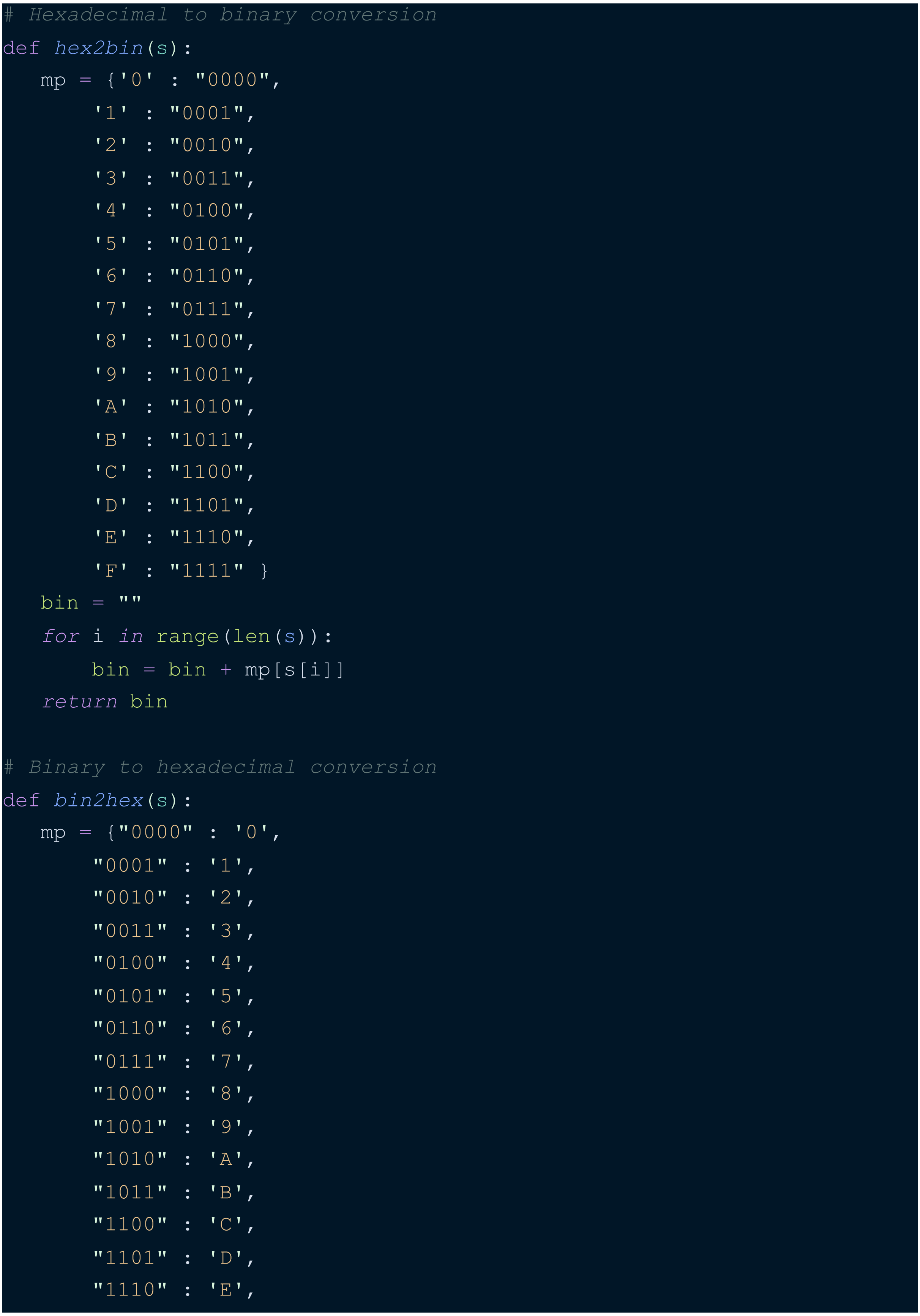
OUTPUT



```
sirkp@sirkp:~/Desktop/Sem 7/IS/lab/CT2$ g++ -std=c++11 miler-robin-primality-test.cpp
sirkp@sirkp:~/Desktop/Sem 7/IS/lab/CT2$ ./a.out
97 4
97 is prime
sirkp@sirkp:~/Desktop/Sem 7/IS/lab/CT2$
```

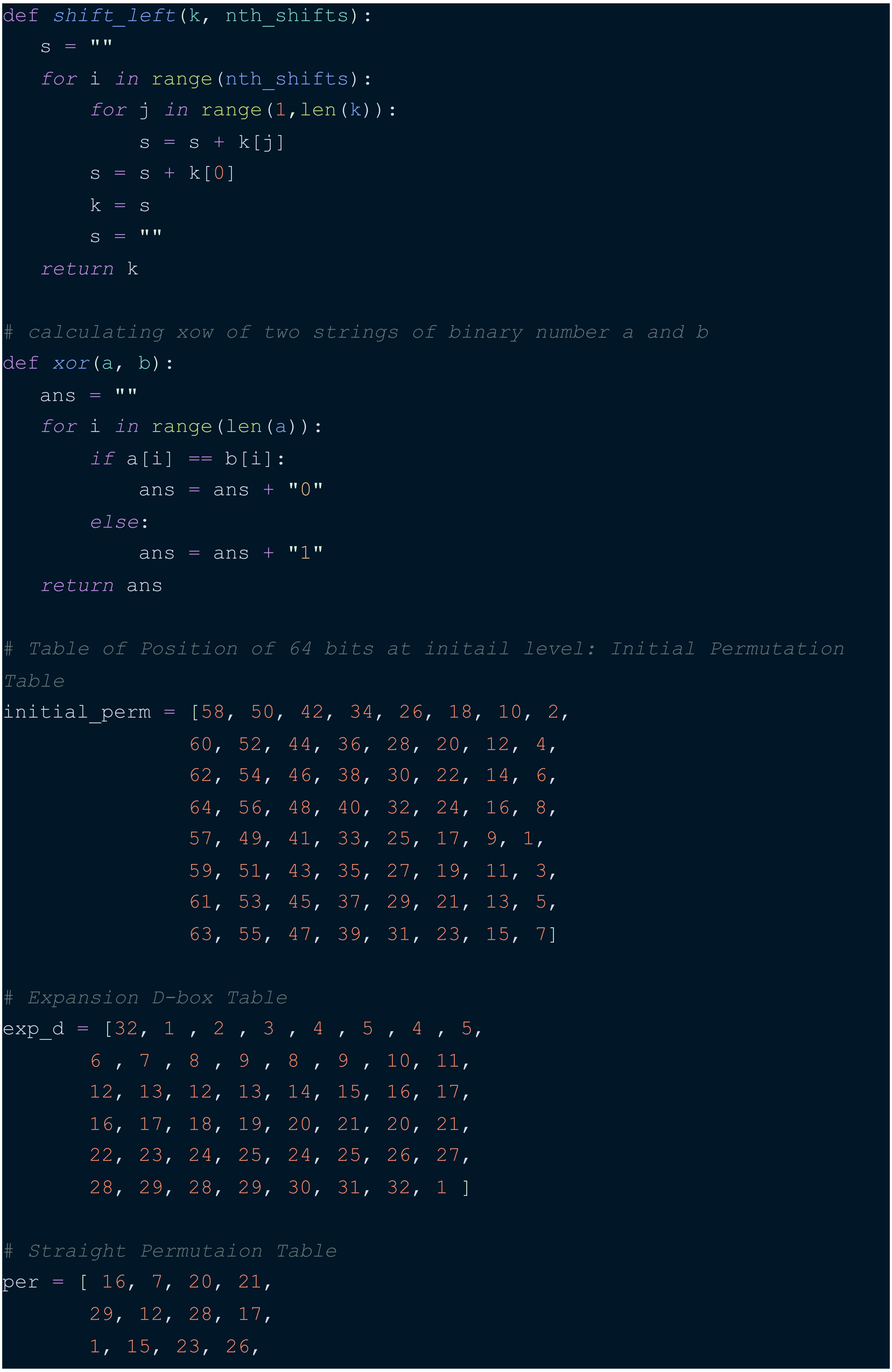
PRACTICAL 9

Write a program for implementation of DES cryptosystem.

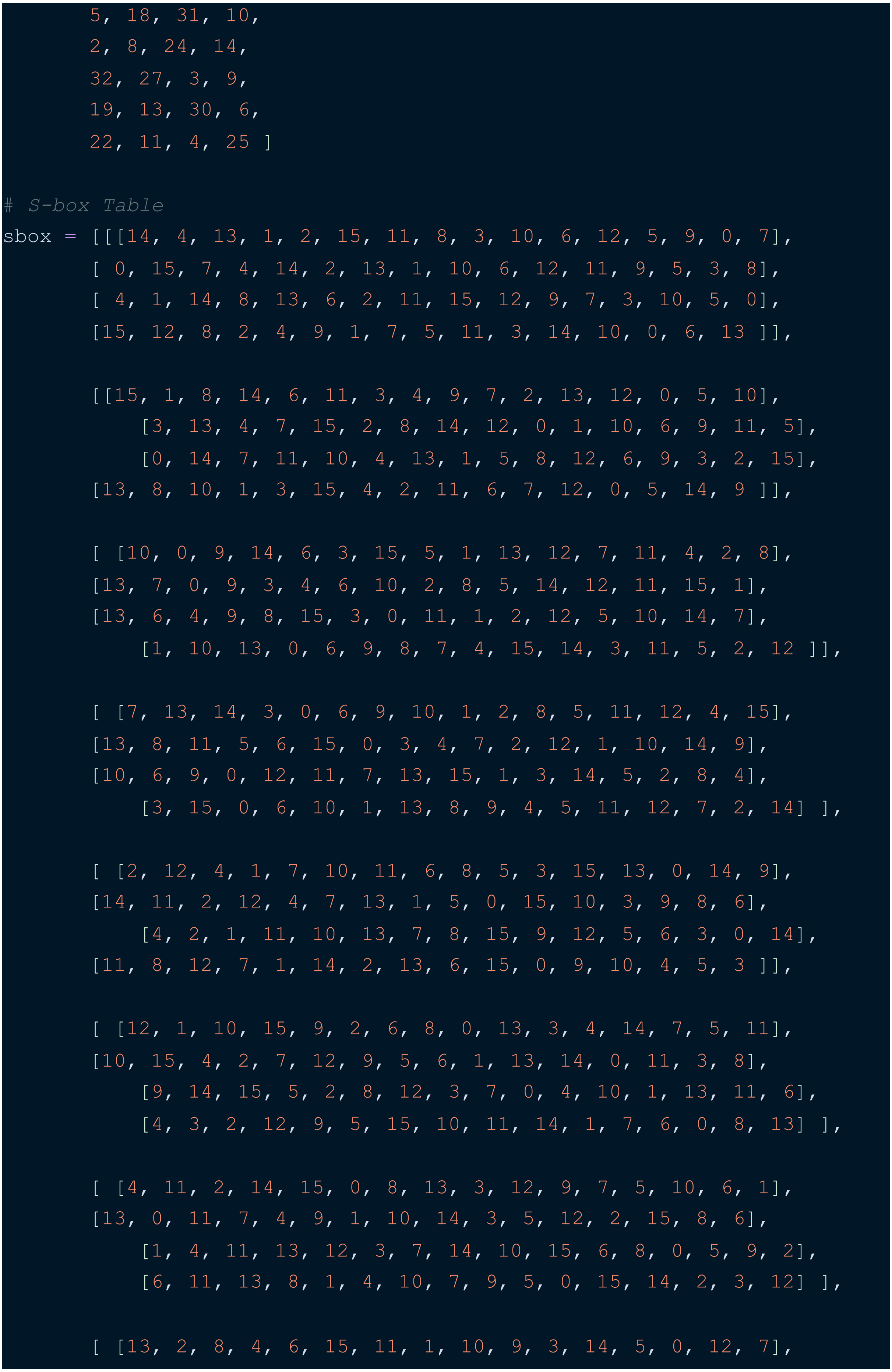


```
# Hexadecimal to binary conversion
def hex2bin(s):
   mp = {'0' : "0000",
        '1' : "0001",
        '2' : "0010",
        '3' : "0011",
        '4' : "0100",
        '5' : "0101",
        '6' : "0110",
        '7' : "0111",
        '8' : "1000",
        '9' : "1001",
        'A' : "1010",
        'B' : "1011",
        'C' : "1100",
        'D' : "1101",
        'E' : "1110",
        'F' : "1111" }
   bin = ""
   for i in range(len(s)):
       bin = bin + mp[s[i]]
   return bin

# Binary to hexadecimal conversion
def bin2hex(s):
   mp = {"0000" : '0',
        "0001" : '1',
        "0010" : '2',
        "0011" : '3',
        "0100" : '4',
        "0101" : '5',
        "0110" : '6',
        "0111" : '7',
        "1000" : '8',
        "1001" : '9',
        "1010" : 'A',
        "1011" : 'B',
        "1100" : 'C',
        "1101" : 'D',
        "1110" : 'E',
```

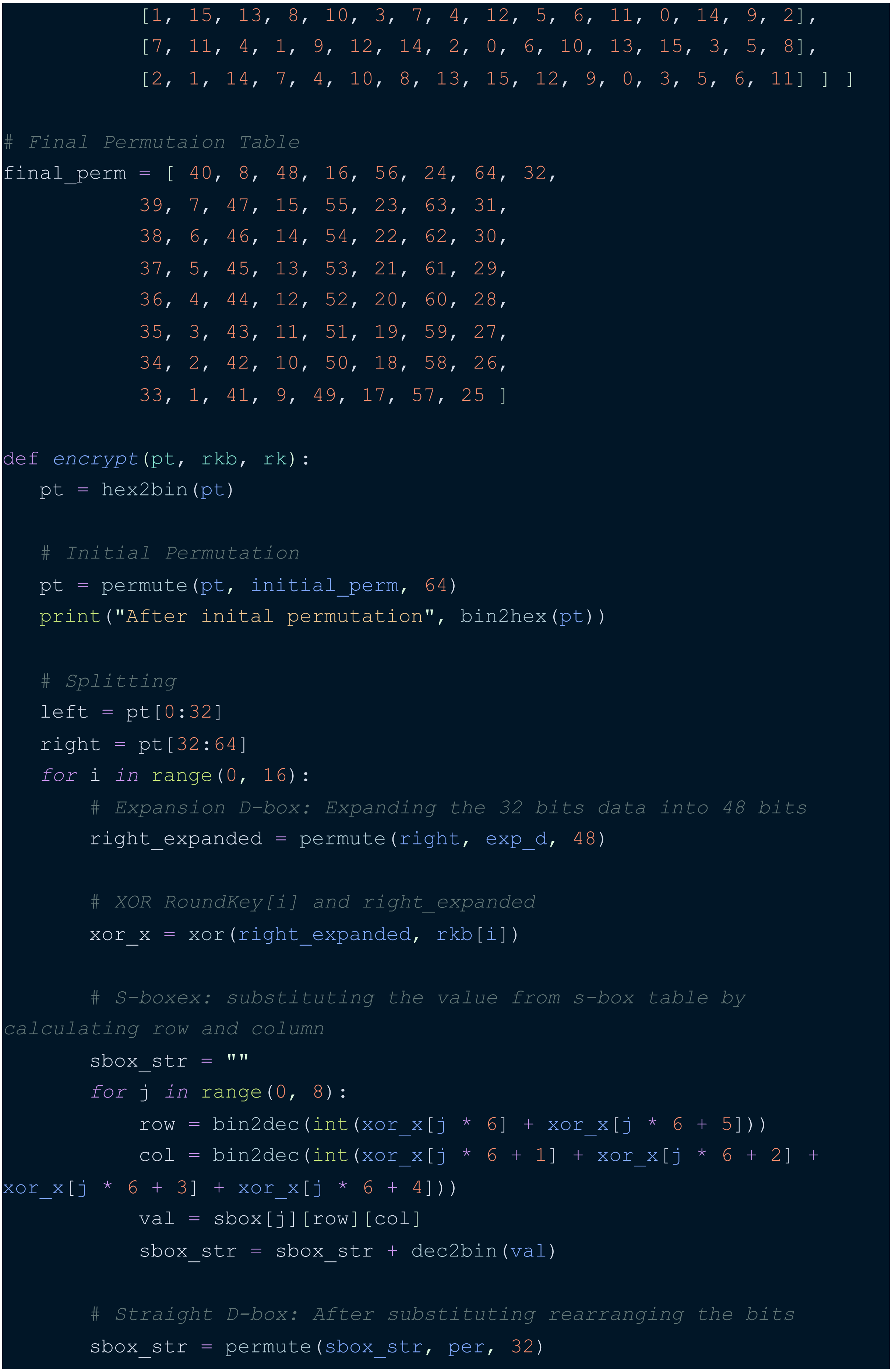
```
        "1111" : 'F' }
    hex = ""
    for i in range(0,len(s),4):
        ch = ""
        ch = ch + s[i]
        ch = ch + s[i + 1]
        ch = ch + s[i + 2]
        ch = ch + s[i + 3]
        hex = hex + mp[ch]

    return hex

# Binary to decimal conversion
def bin2dec(binary):

    binary1 = binary
    decimal, i, n = 0, 0, 0
    while(binary != 0):
        dec = binary % 10
        decimal = decimal + dec * pow(2, i)
        binary = binary//10
        i += 1
    return decimal

# Decimal to binary conversion
def dec2bin(num):
    res = bin(num).replace("0b", "")
    if(len(res)%4 != 0):
        div = len(res) / 4
        div = int(div)
        counter =(4 * (div + 1)) - len(res)
        for i in range(0, counter):
            res = '0' + res
    return res

# Permute function to rearrange the bits
def permute(k, arr, n):
    permutation = ""
    for i in range(0, n):
        permutation = permutation + k[arr[i] - 1]
    return permutation

# shifting the bits towards left by nth shifts
```



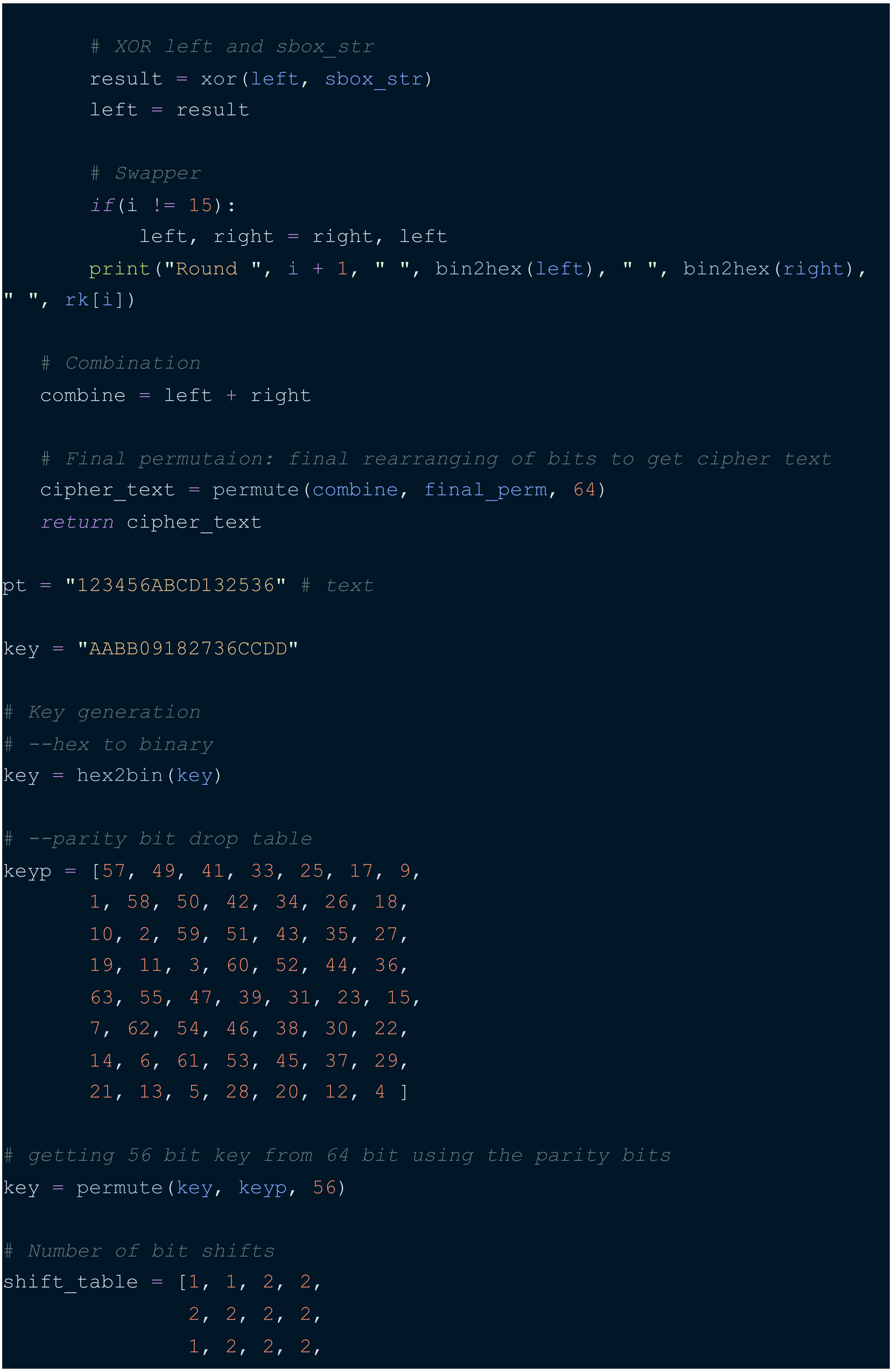
```
def shift_left(k, nth_shifts):
   s = ""
   for i in range(nth_shifts):
       for j in range(1,len(k)):
           s = s + k[j]
       s = s + k[0]
       k = s
       s = ""
   return k

# calculating xow of two strings of binary number a and b
def xor(a, b):
   ans = ""
   for i in range(len(a)):
       if a[i] == b[i]:
           ans = ans + "0"
       else:
           ans = ans + "1"
   return ans

# Table of Position of 64 bits at initail level: Initial Permutation
Table
initial_perm = [58, 50, 42, 34, 26, 18, 10, 2,
               60, 52, 44, 36, 28, 20, 12, 4,
               62, 54, 46, 38, 30, 22, 14, 6,
               64, 56, 48, 40, 32, 24, 16, 8,
               57, 49, 41, 33, 25, 17, 9, 1,
               59, 51, 43, 35, 27, 19, 11, 3,
               61, 53, 45, 37, 29, 21, 13, 5,
               63, 55, 47, 39, 31, 23, 15, 7]

# Expansion D-box Table
exp_d = [32, 1 , 2 , 3 , 4 , 5 , 4 , 5,
       6 , 7 , 8 , 9 , 8 , 9 , 10, 11,
       12, 13, 12, 13, 14, 15, 16, 17,
       16, 17, 18, 19, 20, 21, 20, 21,
       22, 23, 24, 25, 24, 25, 26, 27,
       28, 29, 28, 29, 30, 31, 32, 1 ]

# Straight Permutaion Table
per = [ 16, 7, 20, 21,
       29, 12, 28, 17,
       1, 15, 23, 26,
```



```
       5, 18, 31, 10,
       2, 8, 24, 14,
       32, 27, 3, 9,
       19, 13, 30, 6,
       22, 11, 4, 25 ]

# S-box Table
sbox = [[[14, 4, 13, 1, 2, 15, 11, 8, 3, 10, 6, 12, 5, 9, 0, 7],
       [ 0, 15, 7, 4, 14, 2, 13, 1, 10, 6, 12, 11, 9, 5, 3, 8],
       [ 4, 1, 14, 8, 13, 6, 2, 11, 15, 12, 9, 7, 3, 10, 5, 0],
       [15, 12, 8, 2, 4, 9, 1, 7, 5, 11, 3, 14, 10, 0, 6, 13 ]],

       [[15, 1, 8, 14, 6, 11, 3, 4, 9, 7, 2, 13, 12, 0, 5, 10],
           [3, 13, 4, 7, 15, 2, 8, 14, 12, 0, 1, 10, 6, 9, 11, 5],
           [0, 14, 7, 11, 10, 4, 13, 1, 5, 8, 12, 6, 9, 3, 2, 15],
       [13, 8, 10, 1, 3, 15, 4, 2, 11, 6, 7, 12, 0, 5, 14, 9 ]],

       [ [10, 0, 9, 14, 6, 3, 15, 5, 1, 13, 12, 7, 11, 4, 2, 8],
       [13, 7, 0, 9, 3, 4, 6, 10, 2, 8, 5, 14, 12, 11, 15, 1],
       [13, 6, 4, 9, 8, 15, 3, 0, 11, 1, 2, 12, 5, 10, 14, 7],
           [1, 10, 13, 0, 6, 9, 8, 7, 4, 15, 14, 3, 11, 5, 2, 12 ]],

       [ [7, 13, 14, 3, 0, 6, 9, 10, 1, 2, 8, 5, 11, 12, 4, 15],
       [13, 8, 11, 5, 6, 15, 0, 3, 4, 7, 2, 12, 1, 10, 14, 9],
       [10, 6, 9, 0, 12, 11, 7, 13, 15, 1, 3, 14, 5, 2, 8, 4],
           [3, 15, 0, 6, 10, 1, 13, 8, 9, 4, 5, 11, 12, 7, 2, 14] ],

       [ [2, 12, 4, 1, 7, 10, 11, 6, 8, 5, 3, 15, 13, 0, 14, 9],
       [14, 11, 2, 12, 4, 7, 13, 1, 5, 0, 15, 10, 3, 9, 8, 6],
           [4, 2, 1, 11, 10, 13, 7, 8, 15, 9, 12, 5, 6, 3, 0, 14],
       [11, 8, 12, 7, 1, 14, 2, 13, 6, 15, 0, 9, 10, 4, 5, 3 ]],

       [ [12, 1, 10, 15, 9, 2, 6, 8, 0, 13, 3, 4, 14, 7, 5, 11],
       [10, 15, 4, 2, 7, 12, 9, 5, 6, 1, 13, 14, 0, 11, 3, 8],
           [9, 14, 15, 5, 2, 8, 12, 3, 7, 0, 4, 10, 1, 13, 11, 6],
           [4, 3, 2, 12, 9, 5, 15, 10, 11, 14, 1, 7, 6, 0, 8, 13] ],

       [ [4, 11, 2, 14, 15, 0, 8, 13, 3, 12, 9, 7, 5, 10, 6, 1],
       [13, 0, 11, 7, 4, 9, 1, 10, 14, 3, 5, 12, 2, 15, 8, 6],
           [1, 4, 11, 13, 12, 3, 7, 14, 10, 15, 6, 8, 0, 5, 9, 2],
           [6, 11, 13, 8, 1, 4, 10, 7, 9, 5, 0, 15, 14, 2, 3, 12] ],

       [ [13, 2, 8, 4, 6, 15, 11, 1, 10, 9, 3, 14, 5, 0, 12, 7],
```

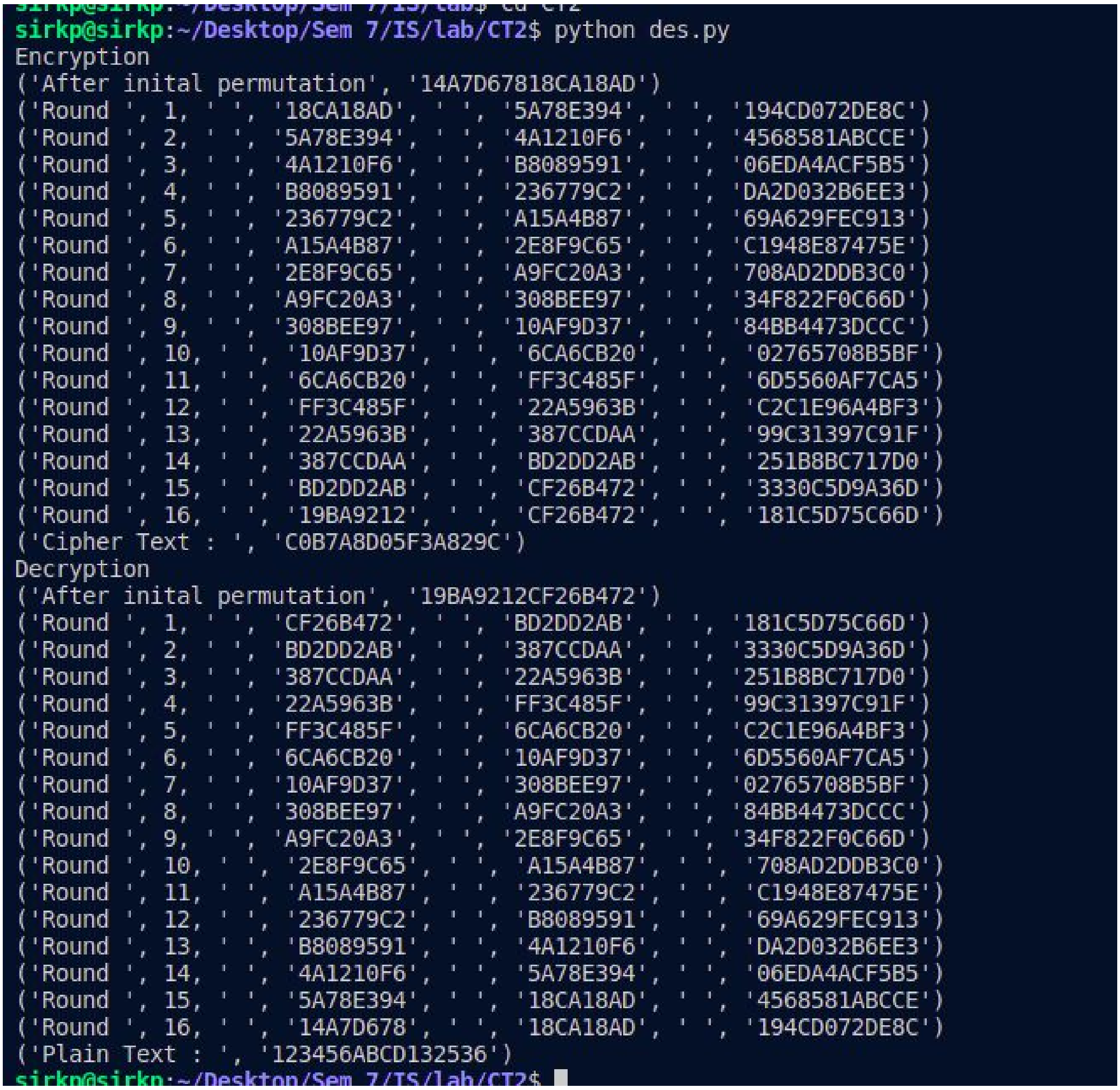


```
           [1, 15, 13, 8, 10, 3, 7, 4, 12, 5, 6, 11, 0, 14, 9, 2],
           [7, 11, 4, 1, 9, 12, 14, 2, 0, 6, 10, 13, 15, 3, 5, 8],
           [2, 1, 14, 7, 4, 10, 8, 13, 15, 12, 9, 0, 3, 5, 6, 11] ] ]

# Final Permutaion Table
final_perm = [ 40, 8, 48, 16, 56, 24, 64, 32,
           39, 7, 47, 15, 55, 23, 63, 31,
           38, 6, 46, 14, 54, 22, 62, 30,
           37, 5, 45, 13, 53, 21, 61, 29,
           36, 4, 44, 12, 52, 20, 60, 28,
           35, 3, 43, 11, 51, 19, 59, 27,
           34, 2, 42, 10, 50, 18, 58, 26,
           33, 1, 41, 9, 49, 17, 57, 25 ]

def encrypt(pt, rkb, rk):
   pt = hex2bin(pt)

   # Initial Permutation
   pt = permute(pt, initial_perm, 64)
   print("After inital permutation", bin2hex(pt))

   # Splitting
   left = pt[0:32]
   right = pt[32:64]
   for i in range(0, 16):
       # Expansion D-box: Expanding the 32 bits data into 48 bits
       right_expanded = permute(right, exp_d, 48)

       # XOR RoundKey[i] and right_expanded
       xor_x = xor(right_expanded, rkb[i])

       # S-boxex: substituting the value from s-box table by
calculating row and column
       sbox_str = ""
       for j in range(0, 8):
           row = bin2dec(int(xor_x[j * 6] + xor_x[j * 6 + 5]))
           col = bin2dec(int(xor_x[j * 6 + 1] + xor_x[j * 6 + 2] +
xor_x[j * 6 + 3] + xor_x[j * 6 + 4]))
           val = sbox[j][row][col]
           sbox_str = sbox_str + dec2bin(val)

       # Straight D-box: After substituting rearranging the bits
       sbox_str = permute(sbox_str, per, 32)
```



```
        # XOR left and sbox_str
        result = xor(left, sbox_str)
        left = result

        # Swapper
        if(i != 15):
            left, right = right, left
        print("Round ", i + 1, " ", bin2hex(left), " ", bin2hex(right),
" ", rk[i])

    # Combination
    combine = left + right

    # Final permutaion: final rearranging of bits to get cipher text
    cipher_text = permute(combine, final_perm, 64)
    return cipher_text

pt = "123456ABCD132536" # text

key = "AABB09182736CCDD"

# Key generation
# --hex to binary
key = hex2bin(key)

# --parity bit drop table
keyp = [57, 49, 41, 33, 25, 17, 9,
       1, 58, 50, 42, 34, 26, 18,
       10, 2, 59, 51, 43, 35, 27,
       19, 11, 3, 60, 52, 44, 36,
       63, 55, 47, 39, 31, 23, 15,
       7, 62, 54, 46, 38, 30, 22,
       14, 6, 61, 53, 45, 37, 29,
       21, 13, 5, 28, 20, 12, 4 ]

# getting 56 bit key from 64 bit using the parity bits
key = permute(key, keyp, 56)

# Number of bit shifts
shift_table = [1, 1, 2, 2,
               2, 2, 2, 2,
               1, 2, 2, 2,
               2, 2, 2, 1 ]

# Key- Compression Table : Compression of key from 56 bits to 48 bits
key_comp = [14, 17, 11, 24, 1, 5,
           3, 28, 15, 6, 21, 10,
           23, 19, 12, 4, 26, 8,
           16, 7, 27, 20, 13, 2,
           41, 52, 31, 37, 47, 55,
           30, 40, 51, 45, 33, 48,
           44, 49, 39, 56, 34, 53,
           46, 42, 50, 36, 29, 32 ]

# Splitting
left = key[0:28] # rkb for RoundKeys in binary
right = key[28:56] # rk for RoundKeys in hexadecimal

rkb = []
rk = []
for i in range(0, 16):
   # Shifting the bits by nth shifts by checking from shift table
   left = shift_left(left, shift_table[i])
   right = shift_left(right, shift_table[i])

   # Combination of left and right string
   combine_str = left + right

   # Compression of key from 56 to 48 bits
   round_key = permute(combine_str, key_comp, 48)

   rkb.append(round_key)
   rk.append(bin2hex(round_key))

print("Encryption")
cipher_text = bin2hex(encrypt(pt, rkb, rk))
print("Cipher Text : ",cipher_text)

print("Decryption")
rkb_rev = rkb[::-1]
rk_rev = rk[::-1]
text = bin2hex(encrypt(cipher_text, rkb_rev, rk_rev))
print("Plain Text : ",text)

# This code is contributed by Aditya Jain
```

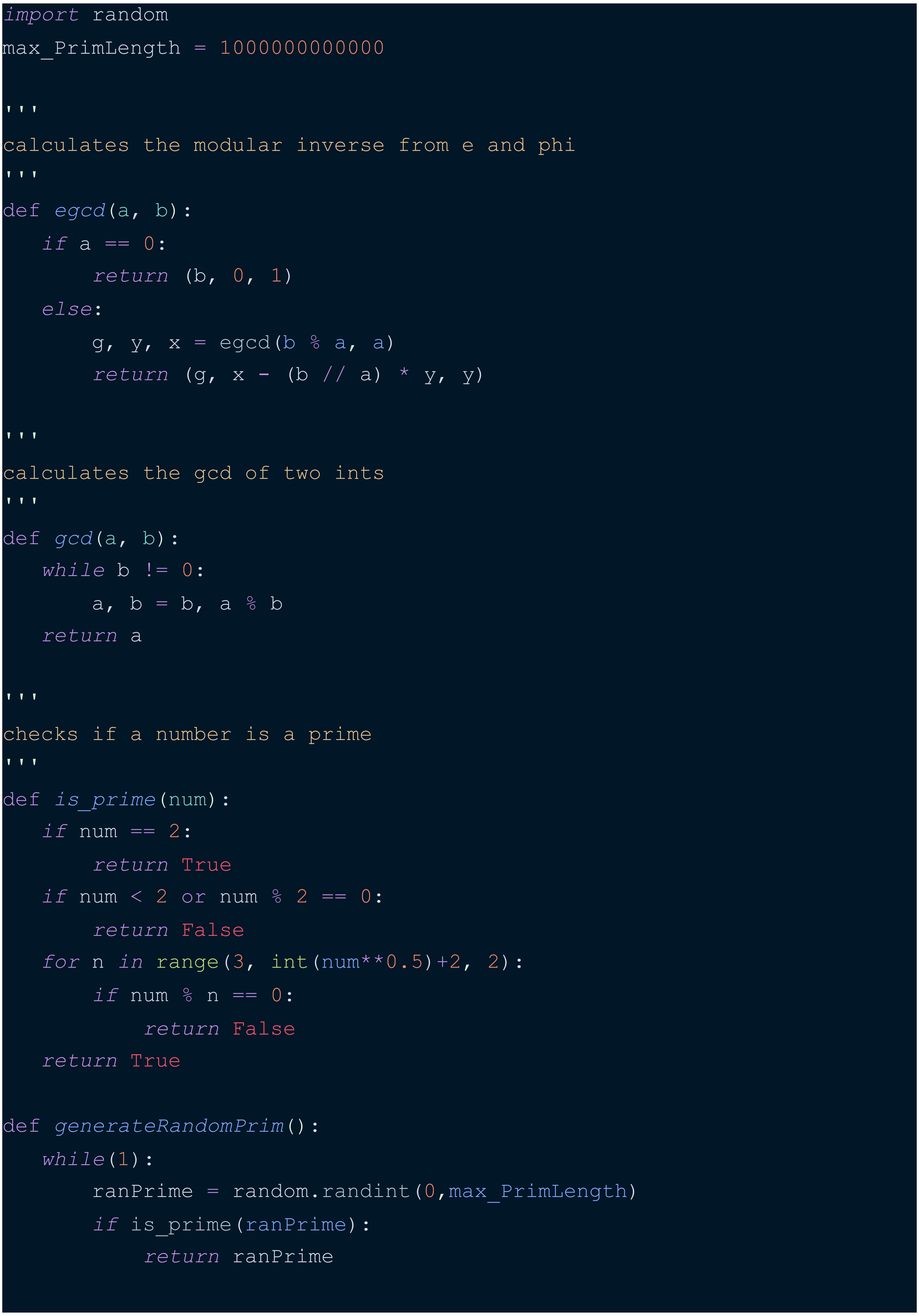
OUTPUT



```
sirkp@sirkp:~/Desktop/Sem 7/IS/lab$ cd CT2
sirkp@sirkp:~/Desktop/Sem 7/IS/lab/CT2$ python des.py
Encryption
('After inital permutation', '14A7D67818CA18AD')
('Round ', 1, ' ', '18CA18AD', ' ', '5A78E394', ' ', '194CD072DE8C')
('Round ', 2, ' ', '5A78E394', ' ', '4A1210F6', ' ', '4568581ABCCE')
('Round ', 3, ' ', '4A1210F6', ' ', 'B8089591', ' ', '06EDA4ACF5B5')
('Round ', 4, ' ', 'B8089591', ' ', '236779C2', ' ', 'DA2D032B6EE3')
('Round ', 5, ' ', '236779C2', ' ', 'A15A4B87', ' ', '69A629FEC913')
('Round ', 6, ' ', 'A15A4B87', ' ', '2E8F9C65', ' ', 'C1948E87475E')
('Round ', 7, ' ', '2E8F9C65', ' ', 'A9FC20A3', ' ', '708AD2DDB3C0')
('Round ', 8, ' ', 'A9FC20A3', ' ', '308BEE97', ' ', '34F822F0C66D')
('Round ', 9, ' ', '308BEE97', ' ', '10AF9D37', ' ', '84BB4473DCCC')
('Round ', 10, ' ', '10AF9D37', ' ', '6CA6CB20', ' ', '02765708B5BF')
('Round ', 11, ' ', '6CA6CB20', ' ', 'FF3C485F', ' ', '6D5560AF7CA5')
('Round ', 12, ' ', 'FF3C485F', ' ', '22A5963B', ' ', 'C2C1E96A4BF3')
('Round ', 13, ' ', '22A5963B', ' ', '387CCDAA', ' ', '99C31397C91F')
('Round ', 14, ' ', '387CCDAA', ' ', 'BD2DD2AB', ' ', '251B8BC717D0')
('Round ', 15, ' ', 'BD2DD2AB', ' ', 'CF26B472', ' ', '3330C5D9A36D')
('Round ', 16, ' ', '19BA9212', ' ', 'CF26B472', ' ', '181C5D75C66D')
('Cipher Text : ', 'C0B7A8D05F3A829C')
Decryption
('After inital permutation', '19BA9212CF26B472')
('Round ', 1, ' ', 'CF26B472', ' ', 'BD2DD2AB', ' ', '181C5D75C66D')
('Round ', 2, ' ', 'BD2DD2AB', ' ', '387CCDAA', ' ', '3330C5D9A36D')
('Round ', 3, ' ', '387CCDAA', ' ', '22A5963B', ' ', '251B8BC717D0')
('Round ', 4, ' ', '22A5963B', ' ', 'FF3C485F', ' ', '99C31397C91F')
('Round ', 5, ' ', 'FF3C485F', ' ', '6CA6CB20', ' ', 'C2C1E96A4BF3')
('Round ', 6, ' ', '6CA6CB20', ' ', '10AF9D37', ' ', '6D5560AF7CA5')
('Round ', 7, ' ', '10AF9D37', ' ', '308BEE97', ' ', '02765708B5BF')
('Round ', 8, ' ', '308BEE97', ' ', 'A9FC20A3', ' ', '84BB4473DCCC')
('Round ', 9, ' ', 'A9FC20A3', ' ', '2E8F9C65', ' ', '34F822F0C66D')
('Round ', 10, ' ', '2E8F9C65', ' ', 'A15A4B87', ' ', '708AD2DDB3C0')
('Round ', 11, ' ', 'A15A4B87', ' ', '236779C2', ' ', 'C1948E87475E')
('Round ', 12, ' ', '236779C2', ' ', 'B8089591', ' ', '69A629FEC913')
('Round ', 13, ' ', 'B8089591', ' ', '4A1210F6', ' ', 'DA2D032B6EE3')
('Round ', 14, ' ', '4A1210F6', ' ', '5A78E394', ' ', '06EDA4ACF5B5')
('Round ', 15, ' ', '5A78E394', ' ', '18CA18AD', ' ', '4568581ABCCE')
('Round ', 16, ' ', '14A7D678', ' ', '18CA18AD', ' ', '194CD072DE8C')
('Plain Text : ', '123456ABCD132536')
sirkp@sirkp:~/Desktop/Sem 7/IS/lab/CT2$
```

PRACTICAL 10

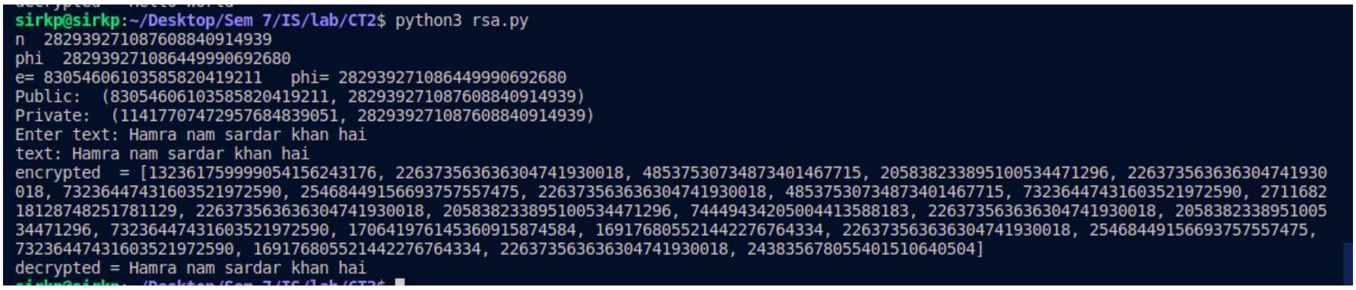
Write a program for implementation of RSA cryptosystem.



```python
import random
max_PrimLength = 1000000000000

'''
calculates the modular inverse from e and phi
'''
def egcd(a, b):
    if a == 0:
        return (b, 0, 1)
    else:
        g, y, x = egcd(b % a, a)
        return (g, x - (b // a) * y, y)

'''
calculates the gcd of two ints
'''
def gcd(a, b):
    while b != 0:
        a, b = b, a % b
    return a

'''
checks if a number is a prime
'''
def is_prime(num):
    if num == 2:
        return True
    if num < 2 or num % 2 == 0:
        return False
    for n in range(3, int(num**0.5)+2, 2):
        if num % n == 0:
            return False
    return True

def generateRandomPrim():
    while(1):
        ranPrime = random.randint(0,max_PrimLength)
        if is_prime(ranPrime):
            return ranPrime
```

```
def generate_keyPairs():
    p = generateRandomPrim()
    q = generateRandomPrim()

    n = p*q
    print("n ",n)
    '''phi(n) = phi(p)*phi(q)'''
    phi = (p-1) * (q-1)
    print("phi ",phi)

    '''choose e coprime to n and 1 > e > phi'''
    e = random.randint(1, phi)
    g = gcd(e,phi)
    while g != 1:
        e = random.randint(1, phi)
        g = gcd(e, phi)

    print("e=",e," ","phi=",phi)
    '''d[1] = modular inverse of e and phi'''
    d = egcd(e, phi)[1]

    '''make sure d is positive'''
    d = d % phi
    if(d < 0):
        d += phi

    return ((e,n),(d,n))

def decrypt(ctext,private_key):
    try:
        key,n = private_key
        text = [chr(pow(char,key,n)) for char in ctext]
        return "".join(text)
    except TypeError as e:
        print(e)

def encrypt(text,public_key):
    key,n = public_key
    ctext = [pow(ord(char),key,n) for char in text]
    return ctext

public_key,private_key = generate_keyPairs()
print("Public: ",public_key)
print("Private: ",private_key)

val = input("Enter text: ")
print("text:", val)

ctext = encrypt(val, public_key)
print("encrypted  =",ctext)
plaintext = decrypt(ctext, private_key)
print("decrypted =",plaintext)
```

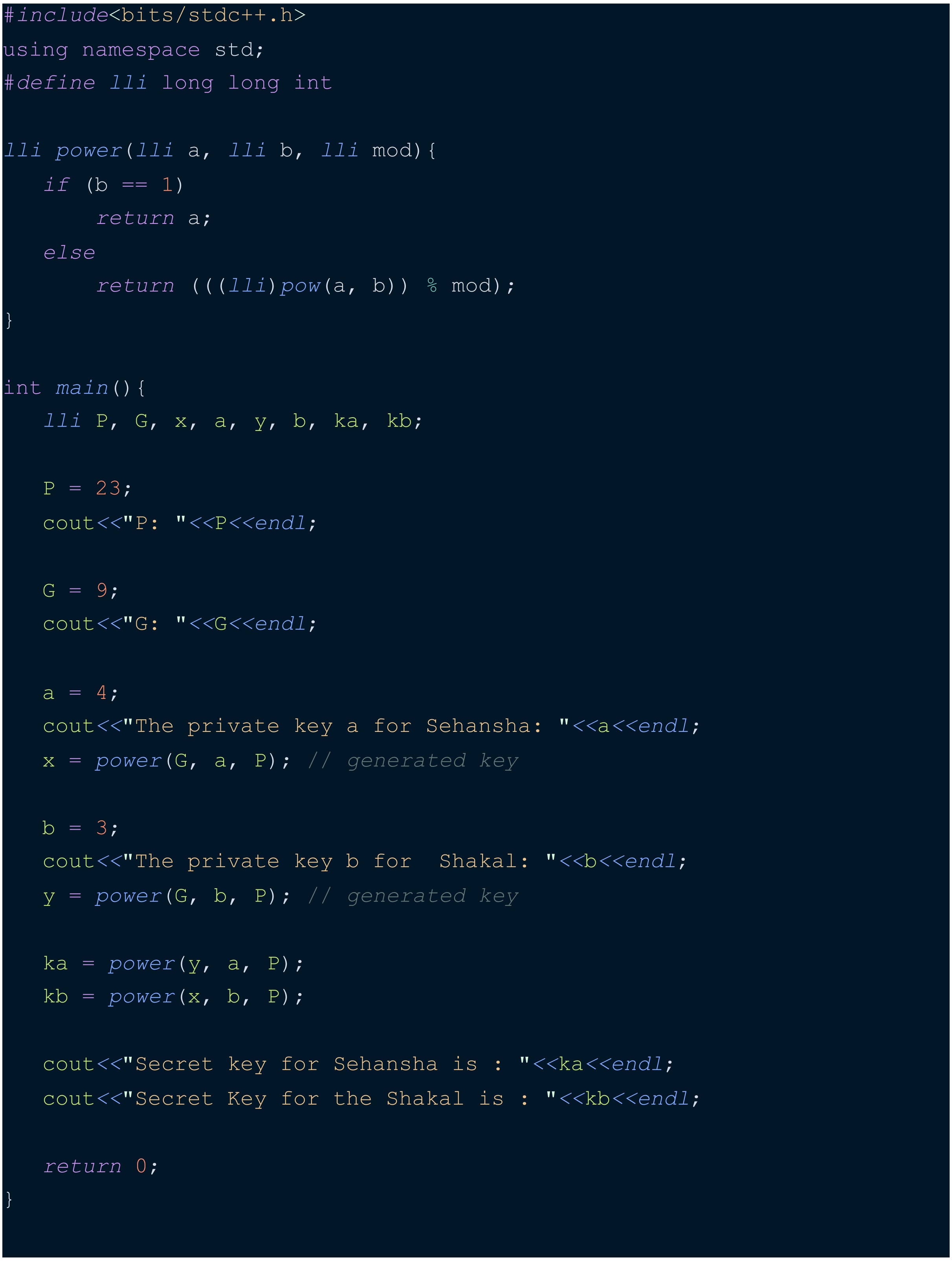
OUTPUT



```
decrypted = Hello World
sirkp@sirkp:~/Desktop/Sem 7/IS/lab/CT2$ python3 rsa.py
n  28293927108760884091 4939
phi  28293927108644999069 2680
e= 83054606103585820419211   phi= 28293927108644999069 2680
Public:  (83054606103585820419211, 28293927108760884091 4939)
Private:  (11417707472957684839051, 28293927108760884091 4939)
Enter text: Hamra nam sardar khan hai
text: Hamra nam sardar khan hai
encrypted  = [13236175999905415624 3176, 22637356363630474193 0018, 48537530734873401467715, 20583823389510053447 1296, 22637356363630474193 0
018, 73236447431603521972590, 25468449156693757557475, 22637356363630474193 0018, 48537530734873401467715, 73236447431603521972590, 2711682
18128748251781129, 22637356363630474193 0018, 20583823389510053447 1296, 74449434205004413588183, 22637356363630474193 0018, 2058382338951005
34471296, 73236447431603521972590, 17064197614536091587 4584, 16917680552144227676 4334, 22637356363630474193 0018, 25468449156693757557475,
73236447431603521972590, 16917680552144227676 4334, 22637356363630474193 0018, 24383567805540151064 0504]
decrypted = Hamra nam sardar khan hai
sirkp@sirkp:~/Desktop/Sem 7/IS/lab/CT2$
```

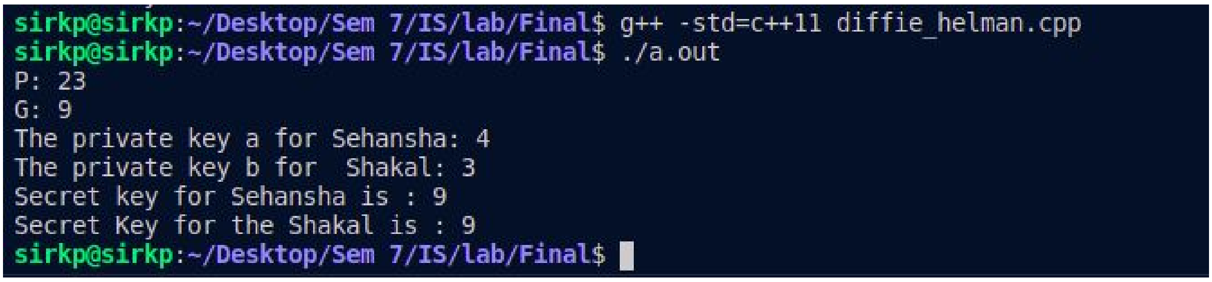
PRACTICAL 11

Write a program for implementation of Diffie Hellman Key Exchange Algorithm.



```
#include<bits/stdc++.h>
using namespace std;
#define lli long long int

lli power(lli a, lli b, lli mod){
   if (b == 1)
       return a;
   else
       return (((lli)pow(a, b)) % mod);
}

int main(){
   lli P, G, x, a, y, b, ka, kb;

   P = 23;
   cout<<"P: "<<P<<endl;

   G = 9;
   cout<<"G: "<<G<<endl;

   a = 4;
   cout<<"The private key a for Sehansha: "<<a<<endl;
   x = power(G, a, P); // generated key

   b = 3;
   cout<<"The private key b for  Shakal: "<<b<<endl;
   y = power(G, b, P); // generated key

   ka = power(y, a, P);
   kb = power(x, b, P);

   cout<<"Secret key for Sehansha is : "<<ka<<endl;
   cout<<"Secret Key for the Shakal is : "<<kb<<endl;

   return 0;
}
```

OUTPUT



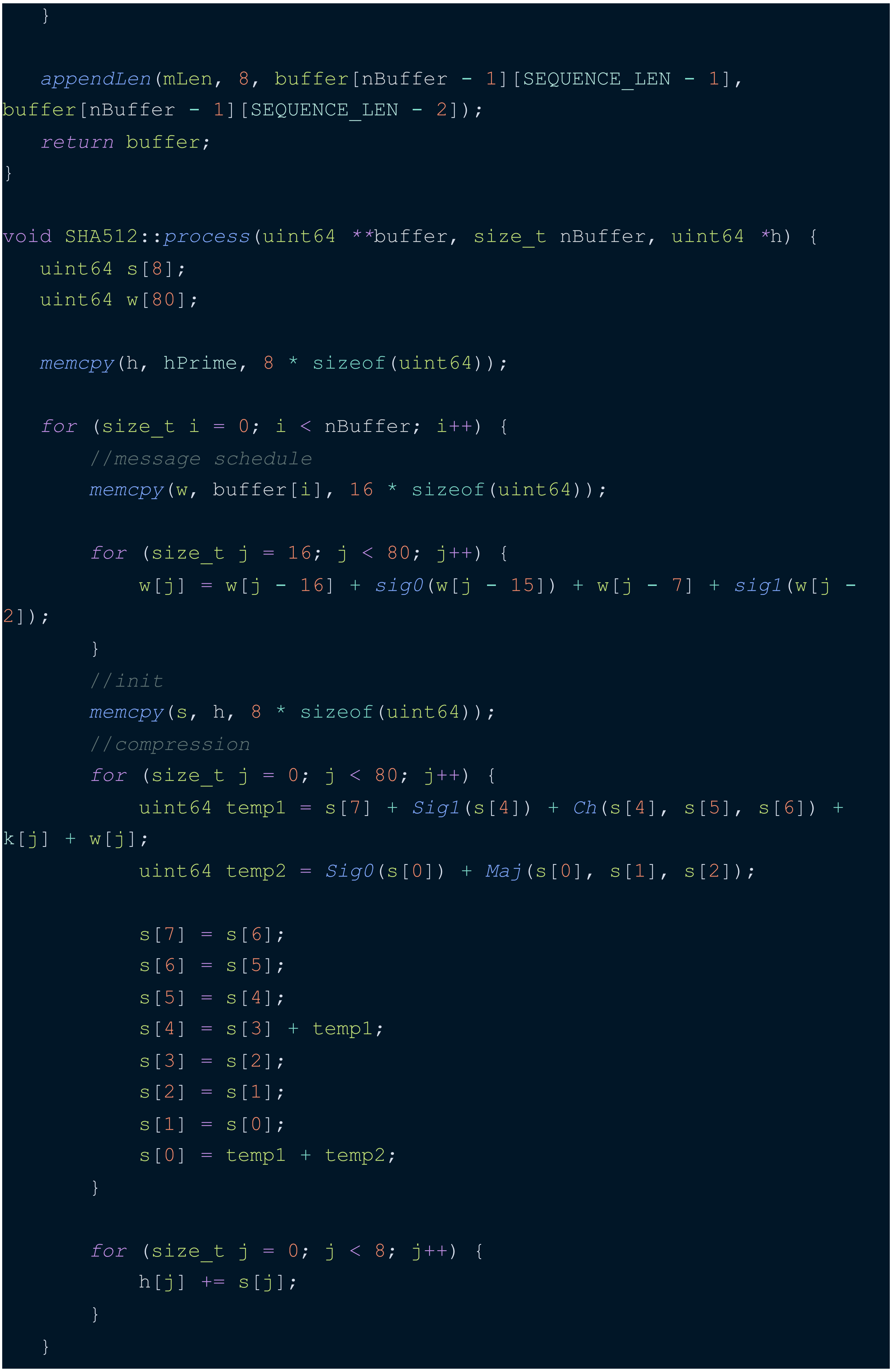
```
sirkp@sirkp:~/Desktop/Sem 7/IS/lab/Final$ g++ -std=c++11 diffie_helman.cpp
sirkp@sirkp:~/Desktop/Sem 7/IS/lab/Final$ ./a.out
P: 23
G: 9
The private key a for Sehansha: 4
The private key b for  Shakal: 3
Secret key for Sehansha is : 9
Secret Key for the Shakal is : 9
sirkp@sirkp:~/Desktop/Sem 7/IS/lab/Final$ 
```

PRACTICAL 12

Write a program for implementation of SHA.

```
#include <stdio.h>
#include <string>
#include <string.h>
#include <iostream>
#include <cstdint>
#include <sstream>
#ifndef H_SHA512
#define H_SHA512
typedef unsigned long long uint64;
#include <string>

using namespace std;

class SHA512 {
private:
   typedef unsigned long long uint64;

   const uint64 hPrime[8] = {0x6a09e667f3bcc908ULL,
                             0xbb67ae8584caa73bULL,
                             0x3c6ef372fe94f82bULL,
                             0xa54ff53a5f1d36f1ULL,
                             0x510e527fade682d1ULL,
                             0x9b05688c2b3e6c1fULL,
                             0x1f83d9abfb41bd6bULL,
                             0x5be0cd19137e2179ULL};

   const uint64 k[80] = {0x428a2f98d728ae22ULL, 0x7137449123ef65cdULL,
0xb5c0fbcfec4d3b2fULL, 0xe9b5dba58189dbbcULL, 0x3956c25bf348b538ULL,
                         0x59f111f1b605d019ULL, 0x923f82a4af194f9bULL,
0xab1c5ed5da6d8118ULL, 0xd807aa98a3030242ULL, 0x12835b0145706fbeULL,
                         0x243185be4ee4b28cULL, 0x550c7dc3d5ffb4e2ULL,
0x72be5d74f27b896fULL, 0x80deb1fe3b1696b1ULL, 0x9bdc06a725c71235ULL,
                         0xc19bf174cf692694ULL, 0xe49b69c19ef14ad2ULL,
0xefbe4786384f25e3ULL, 0x0fc19dc68b8cd5b5ULL, 0x240ca1cc77ac9c65ULL,
                         0x2de92c6f592b0275ULL, 0x4a7484aa6ea6e483ULL,
0x5cb0a9dcbd41fbd4ULL, 0x76f988da831153b5ULL, 0x983e5152ee66dfabULL,
                         0xa831c66d2db43210ULL, 0xb00327c898fb213fULL,
0xbf597fc7beef0ee4ULL, 0xc6e00bf33da88fc2ULL, 0xd5a79147930aa725ULL,
                         0x06ca6351e003826fULL, 0x142929670a0e6e70ULL,
0x27b70a8546d22ffcULL, 0x2e1b21385c26c926ULL, 0x4d2c6dfc5ac42aedULL,
                         0x53380d139d95b3dfULL, 0x650a73548baf63deULL,
0x766a0abb3c77b2a8ULL, 0x81c2c92e47edaee6ULL, 0x92722c851482353bULL,
                         0xa2bfe8a14cf10364ULL, 0xa81a664bbc423001ULL,
0xc24b8b70d0f89791ULL, 0xc76c51a30654be30ULL, 0xd192e819d6ef5218ULL,
                         0xd69906245565a910ULL, 0xf40e35855771202aULL,
0x106aa07032bbd1b8ULL, 0x19a4c116b8d2d0c8ULL, 0x1e376c085141ab53ULL,
                         0x2748774cdf8eeb99ULL, 0x34b0bcb5e19b48a8ULL,
0x391c0cb3c5c95a63ULL, 0x4ed8aa4ae3418acbULL, 0x5b9cca4f7763e373ULL,
                         0x682e6ff3d6b2b8a3ULL, 0x748f82ee5defb2fcULL,
0x78a5636f43172f60ULL, 0x84c87814a1f0ab72ULL, 0x8cc702081a6439ecULL,
                         0x90befffa23631e28ULL, 0xa4506cebde82bde9ULL,
0xbef9a3f7b2c67915ULL, 0xc67178f2e372532bULL, 0xca273eceea26619cULL,
                         0xd186b8c721c0c207ULL, 0xeada7dd6cde0eb1eULL,
0xf57d4f7fee6ed178ULL, 0x06f067aa72176fbaULL, 0x0a637dc5a2c898a6ULL,
                         0x113f9804bef90daeULL, 0x1b710b35131c471bULL,
0x28db77f523047d84ULL, 0x32caab7b40c72493ULL, 0x3c9ebe0a15c9bebcULL,
                         0x431d67c49c100d4cULL, 0x4cc5d4becb3e42b6ULL,
0x597f299cfc657e2aULL, 0x5fcb6fab3ad6faecULL, 0x6c44198c4a475817ULL};

   static const unsigned int SEQUENCE_LEN = (1024 / 64);

   uint64 **preprocess(const unsigned char *input, size_t &nBuffer);
   void appendLen(uint64 mLen, uint64 mp, uint64 &lo, uint64 &hi);
   void process(uint64 **buffer, size_t nBuffer, uint64 *h);
   string digest(uint64 *h);
   void freeBuffer(uint64 **buffer, size_t nBuffer);

public:
   string hash(const string input);

   SHA512();
   ~SHA512();
};

#define Ch(x, y, z) ((x & y) ^ (~x & z))
#define Maj(x, y, z) ((x & y) ^ (x & z) ^ (y & z))
#define RotR(x, n) ((x >> n) | (x << ((sizeof(x) << 3) - n)))
#define Sig0(x) ((RotR(x, 28)) ^ (RotR(x, 34)) ^ (RotR(x, 39)))
#define Sig1(x) ((RotR(x, 14)) ^ (RotR(x, 18)) ^ (RotR(x, 41)))
#define sig0(x) (RotR(x, 1) ^ RotR(x, 8) ^ (x >> 7))
#define sig1(x) (RotR(x, 19) ^ RotR(x, 61) ^ (x >> 6))
#endif
```

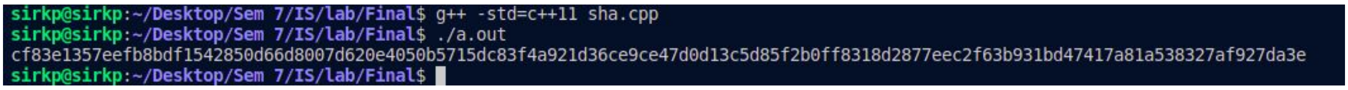
```
SHA512::SHA512() {
}

SHA512::~SHA512() {
}

string SHA512::hash(const string input) {
   size_t nBuffer;  //amt of message blocks
   uint64 **buffer; //message blocks of size 1024bits wtih 16 64bit
words
   uint64 *h = new uint64[8];
   buffer = preprocess((unsigned char *)input.c_str(), nBuffer);
   process(buffer, nBuffer, h);
   freeBuffer(buffer, nBuffer);
   return digest(h);
}

uint64 **SHA512::preprocess(const unsigned char *input, size_t
&nBuffer) {
   size_t mLen = strlen((const char *)input);
   size_t kLen = (895 - (mLen * 8)) % 1024;
   nBuffer = (mLen * 8 + 1 + kLen + 128) / 1024;

   uint64 **buffer = new uint64 *[nBuffer];

   for (size_t i = 0; i < nBuffer; i++) {
       buffer[i] = new uint64[SEQUENCE_LEN];
   }

   for (size_t i = 0; i < nBuffer; i++) {
       for (size_t j = 0; j < SEQUENCE_LEN; j++) {
           uint64 in = 0x0ULL;
           for (size_t k = 0; k < 8; k++) {
               if (i * 128 + j * 8 + k < mLen) {
                   in = in << 8 | (uint64)input[i * 128 + j * 8 + k];
               } else if (i * 128 + j * 8 + k == mLen) {
                   in = in << 8 | 0x80ULL;
               } else {
                   in = in << 8 | 0x0ULL;
               }
           }
           buffer[i][j] = in;
       }
```



```
    }

    appendLen(mLen, 8, buffer[nBuffer - 1][SEQUENCE_LEN - 1],
buffer[nBuffer - 1][SEQUENCE_LEN - 2]);
    return buffer;
}

void SHA512::process(uint64 **buffer, size_t nBuffer, uint64 *h) {
    uint64 s[8];
    uint64 w[80];

    memcpy(h, hPrime, 8 * sizeof(uint64));

    for (size_t i = 0; i < nBuffer; i++) {
        //message schedule
        memcpy(w, buffer[i], 16 * sizeof(uint64));

        for (size_t j = 16; j < 80; j++) {
            w[j] = w[j - 16] + sig0(w[j - 15]) + w[j - 7] + sig1(w[j -
2]);
        }
        //init
        memcpy(s, h, 8 * sizeof(uint64));
        //compression
        for (size_t j = 0; j < 80; j++) {
            uint64 temp1 = s[7] + Sig1(s[4]) + Ch(s[4], s[5], s[6]) +
k[j] + w[j];
            uint64 temp2 = Sig0(s[0]) + Maj(s[0], s[1], s[2]);

            s[7] = s[6];
            s[6] = s[5];
            s[5] = s[4];
            s[4] = s[3] + temp1;
            s[3] = s[2];
            s[2] = s[1];
            s[1] = s[0];
            s[0] = temp1 + temp2;
        }

        for (size_t j = 0; j < 8; j++) {
            h[j] += s[j];
        }
    }
```

```
}

void SHA512::appendLen(uint64 mLen, uint64 mp, uint64 &lo, uint64 &hi)
{
   uint64_t u1 = (mLen & 0xffffffff);
   uint64_t v1 = (mp & 0xffffffff);
   uint64_t t = (u1 * v1);
   uint64_t w3 = (t & 0xffffffff);
   uint64_t k = (t >> 32);
   mLen >>= 32;
   t = (mLen * v1) + k;
   k = (t & 0xffffffff);
   uint64_t w1 = (t >> 32);
   mp >>= 32;
   t = (u1 * mp) + k;
   k = (t >> 32);
   hi = (mLen * mp) + w1 + k;
   lo = (t << 32) + w3;
}

string SHA512::digest(uint64 *h) {
   stringstream ss;
   for (size_t i = 0; i < 8; i++) {
       ss << hex << h[i];
   }
   delete[] h;
   return ss.str();
}

void SHA512::freeBuffer(uint64 **buffer, size_t nBuffer) {
   for (size_t i = 0; i < nBuffer; i++) {
       delete[] buffer[i];
   }

   delete[] buffer;
}

int main(int argc, char *argv[]) {
   SHA512 sha512;
   stringstream ss;
   ss << argv[1];
   cout << sha512.hash(ss.str()) << endl;
   return 0;}
```

OUTPUT



```
sirkp@sirkp:~/Desktop/Sem 7/IS/lab/Final$ g++ -std=c++11 sha.cpp
sirkp@sirkp:~/Desktop/Sem 7/IS/lab/Final$ ./a.out
cf83e1357eefb8bdf1542850d66d8007d620e4050b5715dc83f4a921d36ce9ce47d0d13c5d85f2b0ff8318d2877eec2f63b931bd47417a81a538327af927da3e
sirkp@sirkp:~/Desktop/Sem 7/IS/lab/Final$ 
```