Expt 1

```
import os
array = []
n = int(input("Enter the number of elements : "))
for i in range(0, n):
element = int(input())
array.append(element)
key = int(input("Enter the element for searching :"))
def linearSearch(array, key, size):
for i in range(0, size):
if array[i] == key:
return i
return -1def binarySearch(array, x, low, high):
while low <= high:
mid = low + (high - low)//2
if array[mid] == x:
return mid
elif array[mid] < x:
low = mid + 1
else:
high = mid - 1
return -1
for i in range(2):
pid = os.fork()
if pid == 0:
print("Child process and id is:", os.getpid() , "and it's parent id is:",
os.getppid())
if i == 0:
```

```
print("Linear Search is done by System call 1")
res = linearSearch(array, key, len(array))
print("Element found at index :", res)
elif i == 1:
print("Binary Search is done by System call 2")
res2 = binarySearch(array, key, 0, len(array) - 1)
print("Element found at index :",res2)
break
```

EXPT 2

Child to Child

```
import os


def child1(w):
    os.close(w[0])
    message = input('Enter your message :')
    print()
    os.write(w[1], message.encode())
    os.close(w[1])


def child2(r):
    os.close(r[1])
    message = os.read(r[0], 1024)
    os.close(r[0])
    print("Message recieved by child 2 :", message.decode())


if __name__ == "__main__":
    r, w = os.pipe()
    pid1 = os.fork()
    if pid1 == 0:
        # This is child 1
        print('Process id of child 1 is :', os.getpid())
        print('Parent id of child 1 is :', os.getppid())
        child1((r, w))
        os._exit(0)
    else:
```

```
        # This is the parent
        pid2 = os.fork()
        if pid2 == 0:
            # This is child 2
            child2((r, w))
            print('Process id of child 2 is :', os.getpid())
            print('Parent id of child 2 is :', os.getppid())
            os._exit(0)
        else:
            # This is the parent
            os.close(r)
            os.close(w)
            os.waitpid(pid1, 0)
            os.waitpid(pid2, 0)

###Parent to child

import os

def parent_child_communication():
    r, w = os.pipe()
    pid = os.fork()

    if pid:
        os.close(r)
        msg = input("Enter message :")
        byteString = bytes(msg,'utf-8')
        os.write(w, byteString)
        os.close(w)
```

```
    else:
        os.close(w)
        message = os.read(r, 1024)
        print(f'Child received: {message}')
        os.close(r)


if __name__ == '__main__':
    parent_child_communication()
```

Expt 3

```
###Child to child
import os
import sys

if __name__ == "__main__":
    pipe_fd1 = os.pipe()
    pipe_fd2 = os.pipe()

    pid1 = os.fork()

    if pid1 == 0:
        # This is child 1
        os.close(pipe_fd1[0])
        os.close(pipe_fd2[1])

        while True:
            message = input("Enter a message for Child Process 2: ")
            os.write(pipe_fd1[1], message.encode())

            if message.rstrip() == "STOP":
                print("Communication break by Child Process 1 !!!")
                sys.exit(0)
```

```
        byteMsgFromParent = os.read(pipe_fd2[0], 100)
        msgFromParent = byteMsgFromParent.decode('utf-8')

        if msgFromParent.rstrip() == "STOP":
            os.close(pipe_fd1[1])
            os.close(pipe_fd2[0])
            sys.exit(0)
        else:
            print("Received message from Child Process 2 : ", msgFromParent)

else:
    # This is the parent
    pid2 = os.fork()
    if pid2 == 0:
        # This is child 2
        os.close(pipe_fd1[1])
        os.close(pipe_fd2[0])

    while True:
        byteMsgFromChild = os.read(pipe_fd1[0], 100)
        msgFromChild = byteMsgFromChild.decode('utf-8')

        if msgFromChild.rstrip() == "STOP":
            os.close(pipe_fd1[0])
            os.close(pipe_fd2[1])
            sys.exit(0)
        else:
            print("Received message from Child Process 1: ", msgFromChild)
```

```
        message = input("Enter a message for Child Process 1 : ")
        os.write(pipe_fd2[1], message.encode())

        if message.rstrip() == "STOP":
            print("Communication break by Parent Process 2 !!!")
            sys.exit(0)

####Parent to child
import os
import sys

def main():
    pipe_fd1 = os.pipe()
    pipe_fd2 = os.pipe()

    child_pid = os.fork()

    if child_pid == 0:
        os.close(pipe_fd1[0])
        os.close(pipe_fd2[1])

        while True:
            message = input("Enter a message for parent: ")
            os.write(pipe_fd1[1], message.encode())

            if message.rstrip() == "STOP":
                print("Communication break by Child Process !!!")
                sys.exit(0)
```

```
        byteMsgFromParent = os.read(pipe_fd2[0], 100)
        msgFromParent = byteMsgFromParent.decode('utf-8')

        if msgFromParent.rstrip() == "STOP":
            os.close(pipe_fd1[1])
            os.close(pipe_fd2[0])
            sys.exit(0)
        else:
            print("Received message from parent: ", msgFromParent)

else:
    os.close(pipe_fd1[1])
    os.close(pipe_fd2[0])

    while True:
        byteMsgFromChild = os.read(pipe_fd1[0], 100)
        msgFromChild = byteMsgFromChild.decode('utf-8')

        if msgFromChild.rstrip() == "STOP":
            os.close(pipe_fd1[0])
            os.close(pipe_fd2[1])
            sys.exit(0)
        else:
            print("Received message from child: ", msgFromChild)

        message = input("Enter a message for child: ")
        os.write(pipe_fd2[1], message.encode())

        if message.rstrip() == "STOP":
```

```
            print("Communication break by Parent Process !!!")
            sys.exit(0)

if __name__ == "__main__":
    main()
```

EXPT 4

```
###Process 1
# Process 1
import os
import stat

fifo = "fifo"

os.mkfifo(fifo, stat.S_IRUSR | stat.S_IWUSR)

fd = os.open(fifo, os.O_RDWR)

while True:
    message = input("Enter a message for the process B ('STOP' to quit) : ")

    os.write(fd, message.encode())

    if message.strip() == 'STOP':
        print('Communication Break !!!')
        break
```

```
    message = os.read(fd, 100).decode()
    print("Message received by Process A : ", message)

    if message.strip() == 'STOP':
        print('Communication Break !!!')
        break

os.close(fd)
if os.path.exists(fifo):
    os.unlink(fifo)
#####Process 2
# Process 2
import os
import stat
import time

fifo = "fifo"

if not os.path.exists(fifo):
    os.mkfifo(fifo, stat.S_IRUSR | stat.S_IWUSR)

fd = os.open(fifo, os.O_RDWR)

while True:
    message = os.read(fd, 100).decode()
    print("Message received by Process B : ", message)
    if message.strip() == 'STOP':
        print('Communication Break !!!')
```

```
        break
    message = input("Enter a message for the Process 1 ('STOP' to quit): ")
    os.write(fd, message.encode())
    if message.strip() == 'STOP':
        print('Communication Break !!!')
        break

time.sleep(1)
os.close(fd)
if os.path.exists(fifo):
    os.unlink(fifo)
```

EXPT 5

```
###Process 1
import sysv_ipc
import os

key = sysv_ipc.ftok("keyfile", 65)
mq = sysv_ipc.MessageQueue(key, sysv_ipc.IPC_CREAT)

while True:
    message = input("Enter a message to send (or type 'STOP' to quit): ")

    mq.send(message)

    if message.strip() == 'STOP':
        print("Communication Break !!!")
        break

    message, _ = mq.receive()
```

```
    print("Received reply: ", message.decode())

    if message.decode().strip() == 'STOP':
        print("Communication Break !!!")
        break

mq.remove()
```

###Process 2

```
import sysv_ipc

key = sysv_ipc.ftok("keyfile", 65)
mq = sysv_ipc.MessageQueue(key, sysv_ipc.IPC_CREAT)

while True:
    message, _ = mq.receive(type=1)
    print("Received message: ", message.decode())

    if message.decode().strip() == 'STOP':
        print("Communication Break !!!")
        break

    reply = input("Enter a reply (or type 'STOP' to quit): ")

    mq.send(reply, type=2)
```

```
    if reply.strip() == 'STOP':
        print("Communication Break !!!")
        break
```

EXPT 6

```
####Single client single server
##Client
import socket

def main():
    recevfd = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    server = ('', 5000)
    recevfd.bind(server)

    while True:
        rcv, client = recevfd.recvfrom(1024)
        rcv = rcv.decode()

        if rcv == "stop" or rcv == "STOP":
            print("Communication Break !!!")
```

```
            break

        print("Client : ", rcv)
        snd = input("Enter a message for Sender : ")
        recevfd.sendto(snd.encode(), client)

        if snd == "stop" or snd == "STOP":
            print("Communication Break !!!")
            break

    recevfd.close()

if __name__ == "__main__":
    main()

###Server
import socket

def main():
    sendfd = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    server = ('localhost', 5000)

    while True:
        snd = input("Enter a for the Client : ")
        sendfd.sendto(snd.encode(), server)

        if snd == "stop":
            print("Communication Break !!!")
            break
```

```
        rcv, client = sendfd.recvfrom(1024)
        rcv = rcv.decode()

        if rcv == "stop" or rcv == "STOP":
            print("Communication Break !!!")
            break

        print("Message received by Server : ", rcv)

    sendfd.close()

if __name__ == "__main__":
    main()

####Multiple client multiple server
###Client
import socket

# Client configuration
HOST = '127.0.0.1'
PORT = 5555

# Connect to the server
client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client.connect((HOST, PORT))
print(f"Connected to {HOST}:{PORT}")

try:
```

```
    while True:
        message = input("Enter your message for Server : ")
        client.send(message.encode('utf-8'))
        if message == 'STOP':
            break

        response = client.recv(1024).decode('utf-8')
        print(f"Received from server: {response}")

except KeyboardInterrupt:
    print("Client shutting down.")
    client.send("STOP".encode('utf-8'))

finally:
    client.close()

#####Server
import socket
import threading

HOST = '127.0.0.1'
PORT = 5555

clients = []

def handle_client(client_socket):
    while True:
        data = client_socket.recv(1024).decode('utf-8')
        if data == 'STOP':
```

```
            print(f"Client {client_socket.getpeername()} has requested to STOP.")
            break
        print(f"Received from {client_socket.getpeername()}: {data}")
        response = input("Enter your response: ")
        client_socket.send(response.encode('utf-8'))

    client_socket.close()

# Server setup
server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server.bind((HOST, PORT))
server.listen(5)
print(f"Server listening on {HOST}:{PORT}")

try:
    while True:
        client_socket, addr = server.accept()
        clients.append(client_socket)

        client_handler = threading.Thread(target=handle_client, args=(client_socket,))
        client_handler.start()

except KeyboardInterrupt:
    print("Server shutting down.")
    for client in clients:
        client.send("STOP".encode('utf-8'))
        client.close()
    server.close()
```

EXPT 7

```
####Client
import sysv_ipc, sys, time, os

# Create shared memory object
memory = sysv_ipc.SharedMemory(sysv_ipc.IPC_CREX)

# Write a message to shared memory
memory.write('Hello')

print("Wrote message.")
```

```
# Wait for response
while True:
    message = memory.read()

    if message != 'Hello':
        print("Received response: ", message)
        break

    time.sleep(1)

# Detach shared memory
memory.detach()

# Remove shared memory
memory.remove()
```

####Server

```
import sysv_ipc, sys, time, os

# Get the key of the shared memory segment created by Program 1
key = int(sys.argv[1])

# Create shared memory object
memory = sysv_ipc.SharedMemory(key)

# Read the message from shared memory
```

```
message = memory.read()

print("Read message: ", message)

# Write a response to shared memory
memory.write('World')

print("Wrote response.")

# Detach shared memory
memory.detach()
```

EXPT 8

```
###CLient
import rpyc
import sys

def main():
    c = rpyc.connect("localhost", 18862)
    x = int(sys.argv[1])
    y = int(sys.argv[2])
```

```
    print('Sum of', x ,'&', y, 'is:', c.root.add(x, y))
    print('Difference of', x ,'&', y, 'is:', c.root.sub(x, y))
    print('Product of', x ,'&', y, 'is:', c.root.mul(x, y))
    print('Quotient of', x ,'&', y, 'is:', c.root.div(x, y))

if __name__ == "__main__":
    main()

###Server
import rpyc
from rpyc.utils.server import ThreadedServer


class MyService(rpyc.Service):
    def on_connect(self, conn):
        pass

    def on_disconnect(self, conn):
        pass

    def exposed_add(self, x, y):
        return x + y

    def exposed_sub(self, x, y):
        return x - y

    def exposed_mul(self, x, y):
        return x * y
```

```
    def exposed_div(self, x, y):
        return x / y if y != 0 else 'Error: Division by zero'

if __name__ == "__main__":
    t = ThreadedServer(MyService, port = 18862)
    print('Server started on port 18862')
    t.start()
```