Task 1:
In an online banking application, the home process creates a shared memory which has a structure of:

```
struct shared{
      char sel[100];
      int b;
};
```

The process then creates a pipe. Then it prompts input space for users and provides them to select options of their preferences according to the data given below.
Provide Your Input from Given Options:
1. Type a to Add Money
2. Type w to Withdraw Money
3. Type c to Check Balance
After reading selection from user input, the process stores the user input in the `sel` and sets `b` = `1000` of structure `shared` where `sel` represents user's selected task from the application and `b` represents current balance. Then it prints the selection as "Your selection: `selection`" and sends the data using shared memory and prints about the selection of the user.

After that, home is interrupted and another process which can be denoted by process opr which is a child process of home. Process opr receives and reads data from the shared memory and according to the user input data it executes some operations,

If, user input is "a" then, it provides users an input space to add money and after users enter the amount to be added it adds the amount with the current balance which is stored in `b` field of structure `shared` if the amount entered to be added by the user is a positive value greater than 0. After successful addition, it prints the acknowledgement of the operation and updated balance after addition. If the entered amount by the user is less than or equal 0 then it will print "Adding failed, Invalid amount".

If, user input is "w" then, it provides users an input space to withdraw money and after users enter the amount to be withdrawn it withdraws the amount from the current balance which is stored in `b` field of structure `shared` if the amount entered to be withdrawn by the user is less than the current balance and a positive value greater than 0. After successful withdrawal, it prints the acknowledgement of the operation and updated balance after withdrawal. If the entered amount by the user is less than or equal 0 or it is less than the current balance then it will print "Withdrawal failed, Invalid amount".

If, user input is "c" then, it prints the current balance which is stored in `b` field of structure `shared`.

For any other inputs opr process prints "Invalid selection".

Finally before termination the process opr writes "Thank you for using" in the pipe and it terminates.

After the termination of its child opr, process home continues and reads the data from the pipe which was written by its child process opr and prints it. Then it terminates by removing the shared memory.

Provide a proper solution by executing two cooperating processes according to the flow of processes described in the scenario given above, ensuring proper communication between processes using shared memory.

Sample Output in the Terminal:

Execution Command in Terminal: ./p1

```
Provide Your Input From Given Options:
1. Type a to Add Money
2. Type w to Withdraw Money
3. Type c to Check Balance
a
Your selection: a

Enter amount to be added:
100
Balance added successfully
Updated balance after addition:
1100
Thank you for using
```

Execution Command in Terminal: ./p1

```
Provide Your Input From Given Options:
1. Type a to Add Money
2. Type w to Withdraw Money
3. Type c to Check Balance
a
Your selection: a

Enter amount to be added:
-600
Adding failed, Invalid amount
Thank you for using
```

Execution Command in Terminal: ./p1

```
Provide Your Input From Given Options:
1. Type a to Add Money
```

2. Type w to Withdraw Money

3. Type c to Check Balance

w

Your selection: w

Enter amount to be withdrawn:

600

Balance withdrawn successfully

Updated balance after withdrawal:

400

Thank you for using

Execution Command in Terminal: ./p1

Provide Your Input From Given Options:

1. Type a to Add Money

2. Type w to Withdraw Money

3. Type c to Check Balance

w

Your selection: w

Enter amount to be withdrawn:

-300

Withdrawal failed, Invalid amount

Thank you for using

Execution Command in Terminal: ./p1

Provide Your Input From Given Options:

1. Type a to Add Money

2. Type w to Withdraw Money

3. Type c to Check Balance

c

Your selection: c

Your current balance is:

1000

Thank you for using

Execution Command in Terminal: ./p1

Provide Your Input From Given Options:

1. Type a to Add Money

2. Type w to Withdraw Money

3. Type c to Check Balance

o

Your selection: o

Invalid selection

Thank you for using

Task 2:

An office management platform executes its log in process first. Upon its execution log in creates a message queue of structure:

```
struct msg{
  long int type;
  char txt[6];
};
```

The process takes input from an user about his/her workspace name. If the workspace is not “cse321”, then it prints “Invalid workspace name”. If the workspace name is “cse321” then it writes the name of the workspace in `txt and` sets the `type` of message queue using a positive integer and prints about its written message in the queue. Then the process gets interrupted, and another process denoted as otp generator which is a child process of log in starts running.

Otp generator reads the first message with the `type` set by its parent process and prints its read message from the queue. Then it generates its own process id as otp and stores it in `txt` and sets a different `type` in the message queue and prints about its stored message in the queue. The destination of this message is log in process.

Otp generator again sends the same message to another process mail which is its child process but this time by assigning a different `type.` Then it prints the message which is being sent to process mail. Then it gets interrupted, and its child process mail starts running.

Process mail reads the first message with the `type` set by its parent process and prints its received message from the queue. It then writes the same message which was read by it in the queue and sends it to process log in by setting a different `type` and prints about its sent message. Then it terminates and its parent otp generator continues executing and it terminates as well.

After the termination of its child, log in continues running and immediately receives the message sent from its child process from the message queue using the mechanism of reading first message with the same `type` which was set by otp generator during sending the message and after reading, the process prints the message.

Then log in again reads the message sent from mail by using the mechanism of reading first message with the same `type` which was set by mail and prints the message after reading as well.

Lastly, log in compares messages received from otp generator and mail. If both messages are the same, it prints “OTP Verified”. Otherwise, it prints “OTP Incorrect”. Then it terminates by removing the message queue.

Provide a proper solution by executing three cooperating processes according to the flow of processes described in the scenario given above, ensuring proper communication between processes using message queue. (OTP will be different for every execution as every time a process gets executed it is assigned a new id).

Sample Output in the Terminal:

Execution Command in Terminal: ./p1

```
Please enter the workspace name:
cse321
Workspace name sent to otp generator from log in: cse321

OTP generator received workspace name from log in: cse321

OTP sent to log in from OTP generator: 13055
OTP sent to mail from OTP generator: 13055
Mail received OTP from OTP generator: 13055
OTP sent to log in from mail: 13055
Log in received OTP from OTP generator: 13055
Log in received OTP from mail: 13055
OTP Verified
```

Execution Command in Terminal: ./p1

```
Please enter the workspace name:
abc
Invalid workspace name
```