**Banking System**

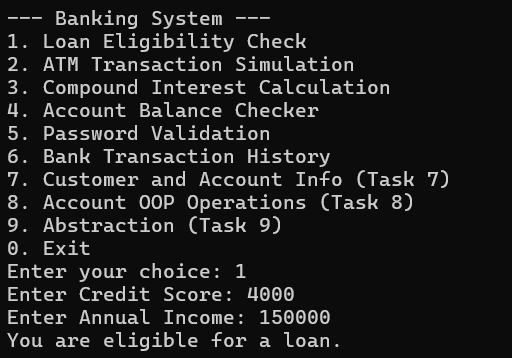
**Control Structure**

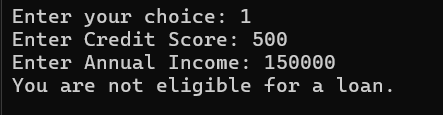
**Task 1:** Conditional Statements In a bank, you have been given the task is to create a program that checks if a customer is eligible for a loan based on their credit score and income. The eligibility criteria are as follows: • Credit Score must be above 700. • Annual Income must be at least $50,000.

Tasks: 1. Write a program that takes the customer's credit score and annual income as input.

2. Use conditional statements (if-else) to determine if the customer is eligible for a loan.

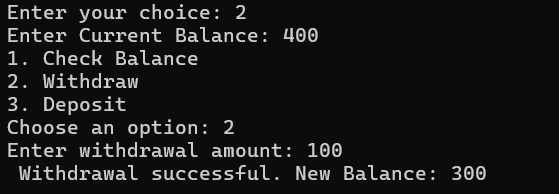
3. Display an appropriate message based on eligibility.

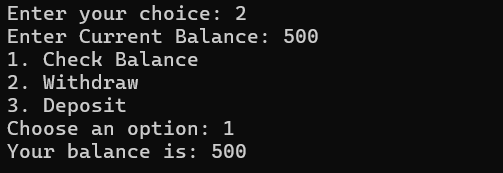


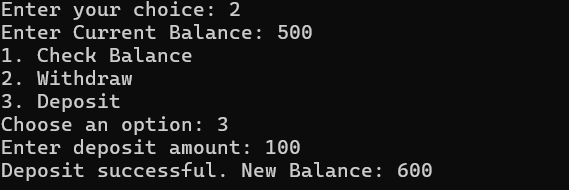


**Task 2: Nested Conditional Statements**

Create a program that simulates an ATM transaction. Display options such as "Check Balance," "Withdraw," "Deposit,". Ask the user to enter their current balance and the amount they want to withdraw or deposit. Implement checks to ensure that the withdrawal amount is not greater than the available balance and that the withdrawal amount is in multiples of 100 or 500. Display appropriate messages for success or failure.







**Task 3: Loop Structures**

You are responsible for calculating compound interest on savings accounts for bank customers. You need to calculate the future balance for each customer's savings account after a certain number of years.

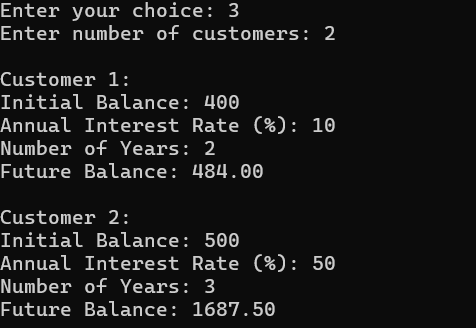
Tasks: 1. Create a program that calculates the future balance of a savings account.

2. Use a loop structure (e.g., for loop) to calculate the balance for multiple customers.

3. Prompt the user to enter the initial balance, annual interest rate, and the number of years.

4. Calculate the future balance using the formula: future\_balance = initial\_balance \* (1 + annual\_interest\_rate/100)^years.

5. Display the future balance for each customer.



**Task 4: Looping, Array and Data Validation**

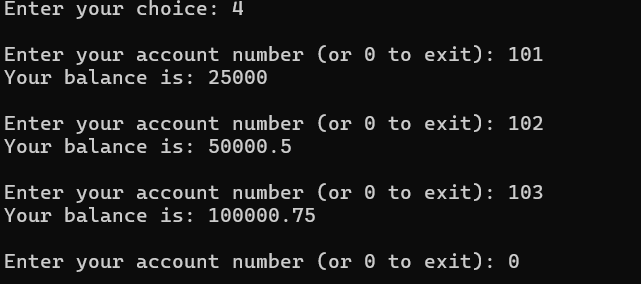
You are tasked with creating a program that allows bank customers to check their account balances. The program should handle multiple customer accounts, and the customer should be able to enter their account number, balance to check the balance.

Tasks: 1. Create a Python program that simulates a bank with multiple customer accounts.

2. Use a loop (e.g., while loop) to repeatedly ask the user for their account number and balance until they enter a valid account number.

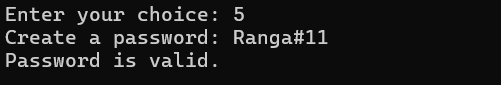
3. Validate the account number entered by the user.

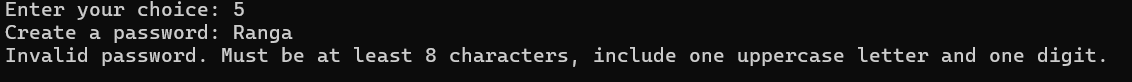
4. If the account number is valid, display the account balance. If not, ask the user to try again.



**Task 5: Password Validation**

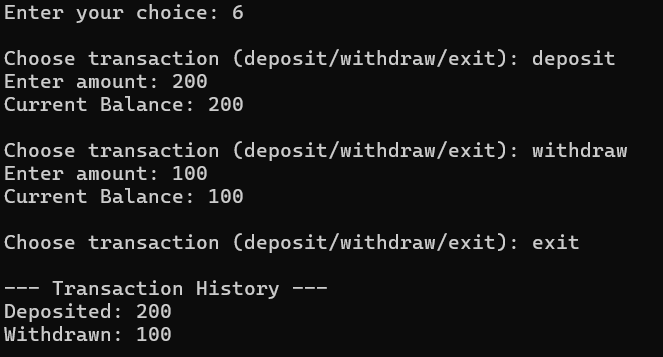
Write a program that prompts the user to create a password for their bank account. Implement if conditions to validate the password according to these rules: • The password must be at least 8 characters long. • It must contain at least one uppercase letter. • It must contain at least one digit. • Display appropriate messages to indicate whether their password is valid or not.





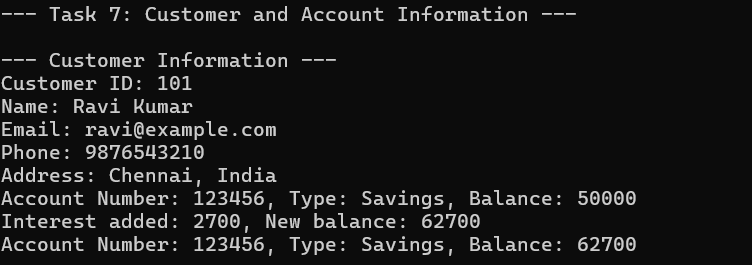
**Task 6: Password Validation**

Create a program that maintains a list of bank transactions (deposits and withdrawals) for a customer. Use a while loop to allow the user to keep adding transactions until they choose to exit. Display the transaction history upon exit using looping statements.

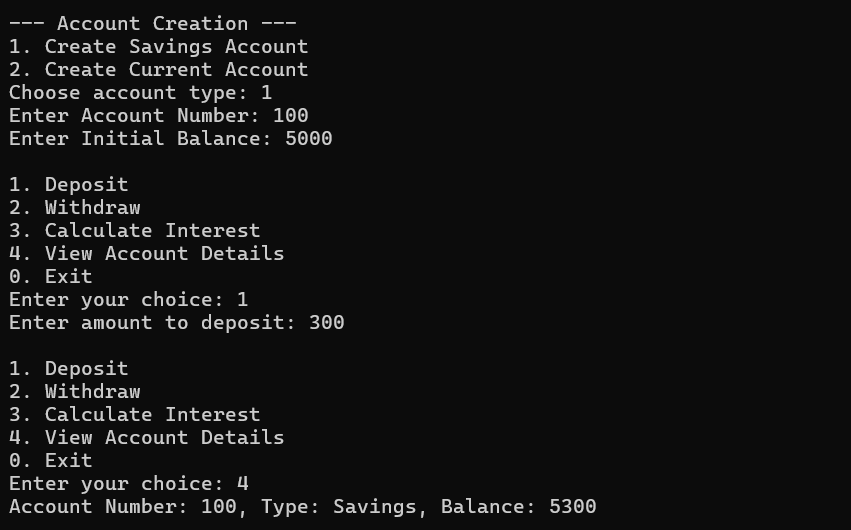


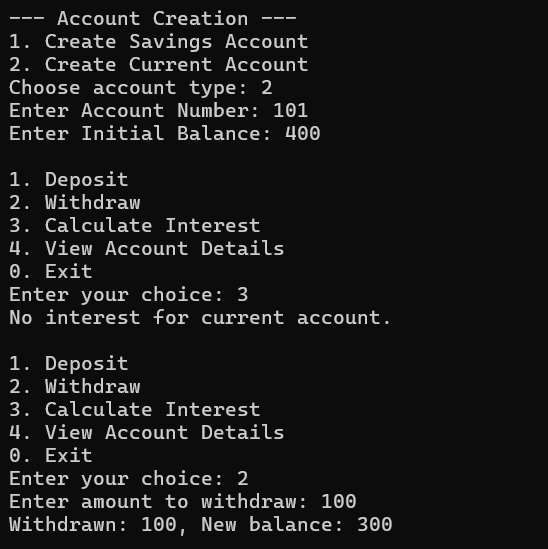
**OOPS, Collections and Exception Handling**

Task 7: Class & Object

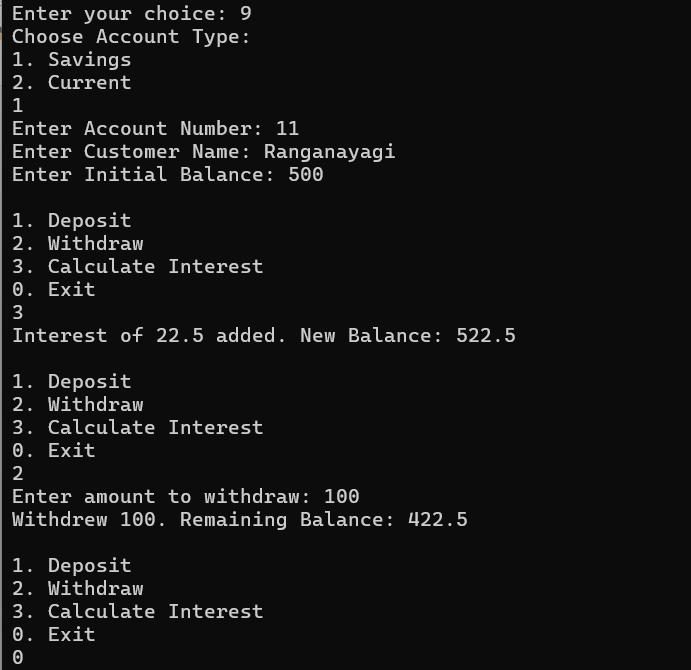


**Task 8: Inheritance and polymorphism**

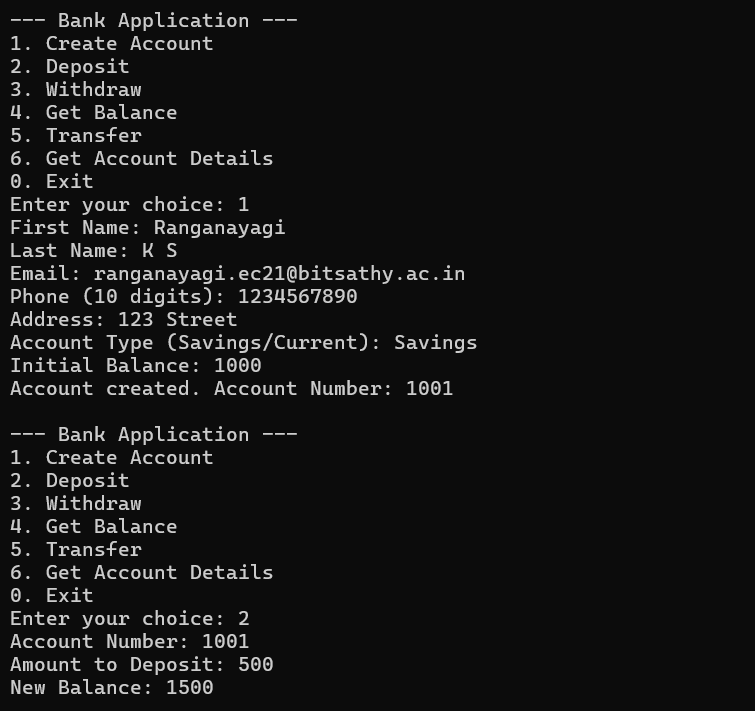


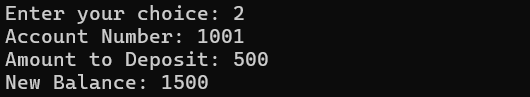


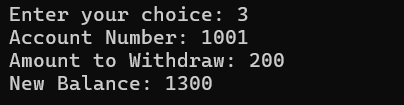
**Task 9: Abstraction**

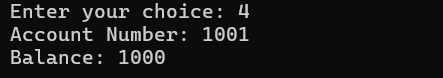


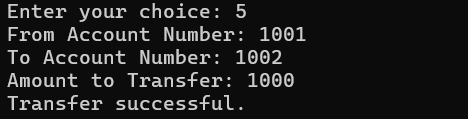
**Task 10: Has A Relation / Association**

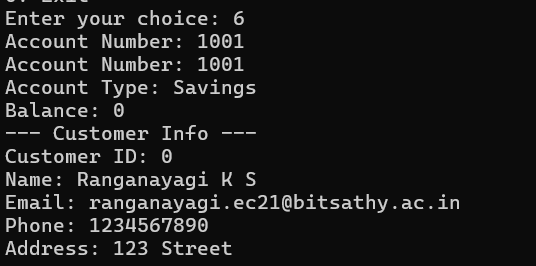




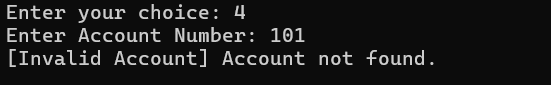


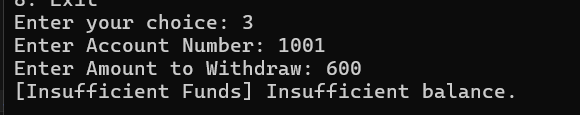






**Task 11-13**

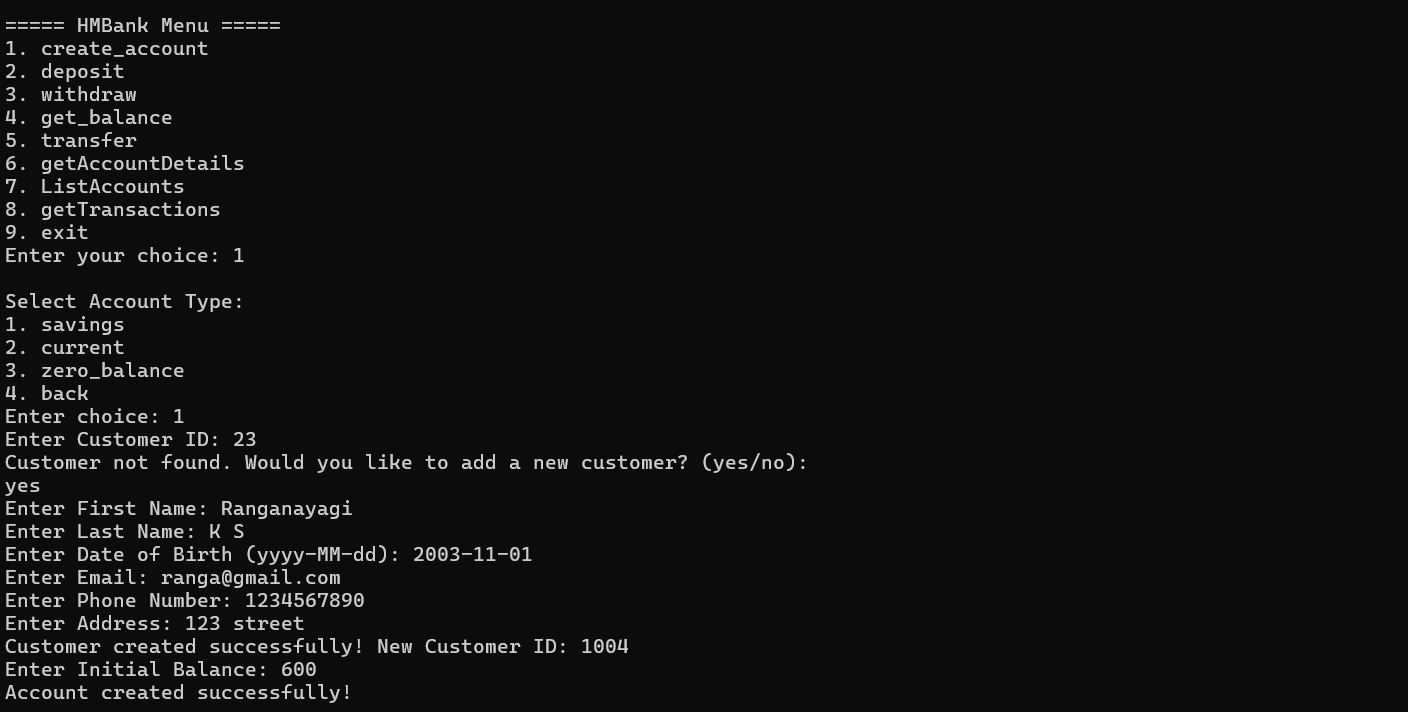
****

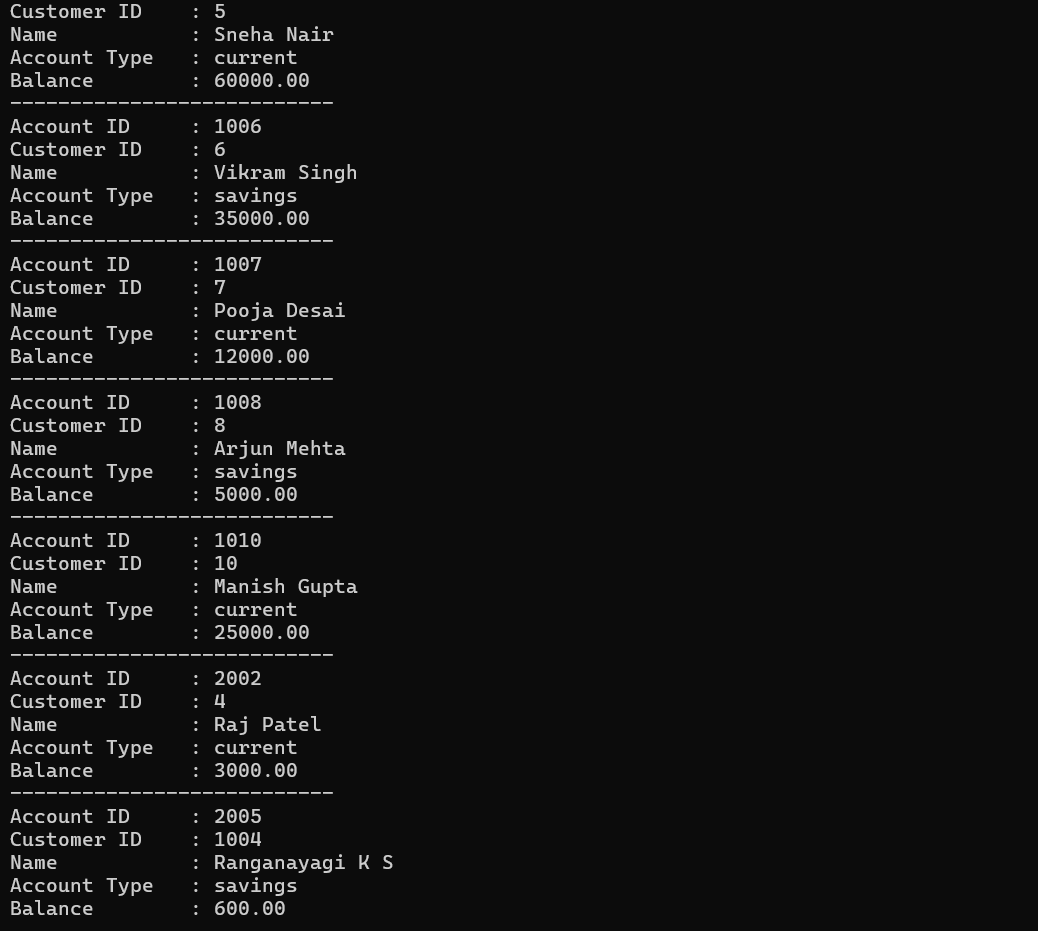
****

****

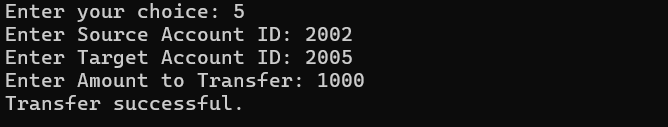
**Task 14 – Database Connection**

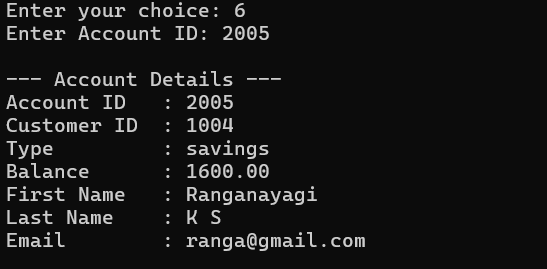
**Create Account**



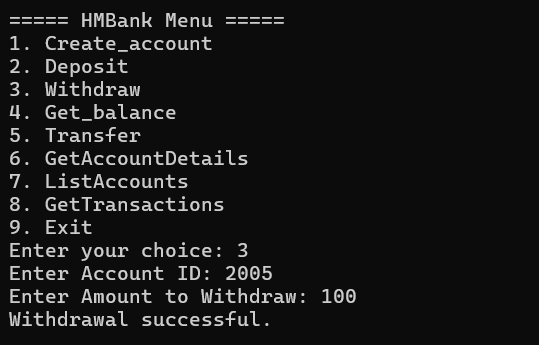
****

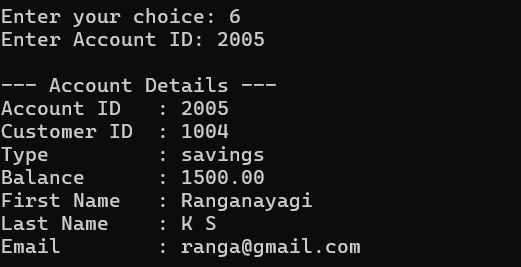
**Transfer Amount**

****

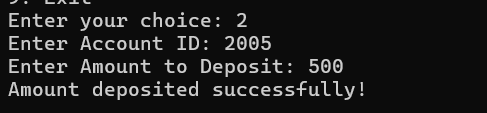
****

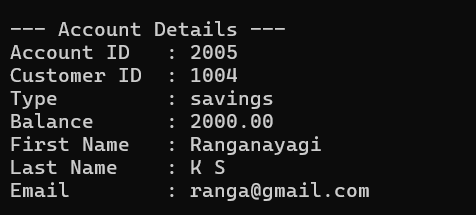
**Withdraw Amount**

****

****

**Deposit Amount**

****

****