PRACTICAL 1

AIM : A program that models a bank account, with classes for the account, the customer, and the bank.

Problem Statement:

Create a program that models a bank account system. The system should consist of classes representing the bank account, the customer, and the bank itself

.

Program Description:

The program will simulate basic banking operations such as creating a new account, depositing and withdrawing money, checking the account balance, and managing customer information. It will use object-oriented programming principles with classes to organize and structure the data and behavior.

Algorithm:

1. Define a **Customer** class with attributes like customer ID, name, address, and contact details.
2. Create a **BankAccount** class with attributes such as account number, account holder (linked to a **Customer**), balance, and account type (e.g., savings or checking).
3. Implement methods in the **BankAccount** class for deposit, withdrawal, and checking the account balance.
4. Develop a **Bank** class to manage a collection of accounts, allowing the creation of new accounts, retrieval of account details, and overall management of the bank.

Source Code:

```
import random

class Customer:
    def __init__(self, name, address, contact_number):
        self.name = name
        self.address       =
        address
        self.contact_number = contact_number
        self.accounts = []
```

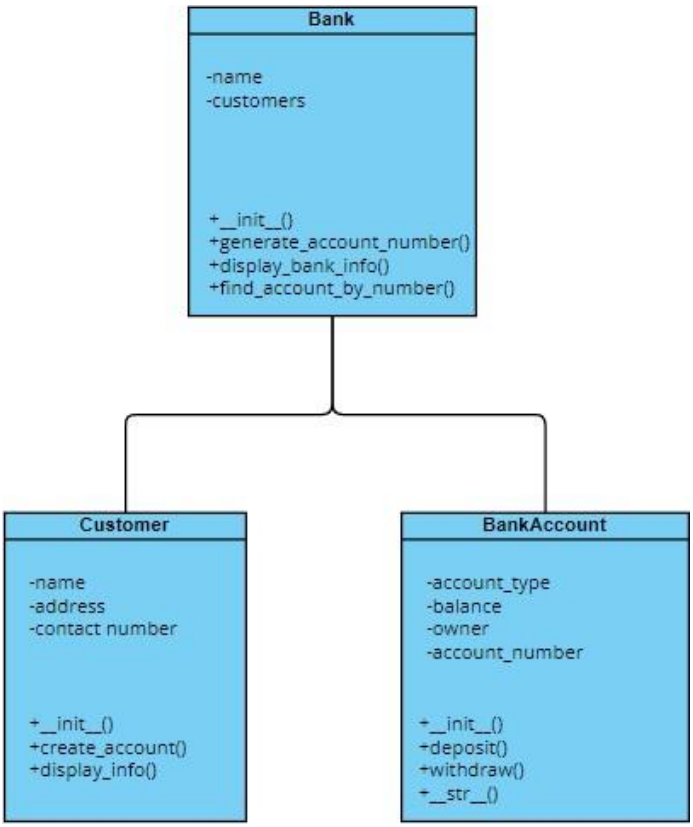
```
    def create_account(self, account_type, initial_balance):
        account_number = Bank.generate_account_number() account =
        BankAccount(account_type, initial_balance, self,
        account_number) self.accounts.append(account) return account

    def        display_customer_info(self):
        print(f"Customer                Name:
        {self.name}")         print(f"Address:
        {self.address}")
        print(f"Contact      Number:      {self.contact_number}")
        print("Accounts:")

        for account in self.accounts:
            print(f" - {account}")

class BankAccount:
    def___init__(self,   account_type,   balance,   owner,   account_number):
        self.account_type = account_type
        self.balance           =
        balance self.owner =
        owner
        self.account_number = account_number

    def deposit(self, amount):
        self.balance += amount
        print(f"Deposited INR {amount}. New balance: INR {self.balance}")

    def withdraw(self, amount):
        if amount <= self.balance:
            self.balance -= amount
            print(f"Withdrew INR {amount}. New balance: INR {self.balance}")
        else:
            print("Insufficient funds!")

    def_str__(self):
        return f"{self.account_type} Account - Account Number: {self.account_number}, Balance: INR
{self.balance}"

class Bank:
    def_init__(self, name):
        self.name = name
        self.customers = []

    def                add_customer(self,                customer):
        self.customers.append(customer)
```

```
    @staticmethod    def
    generate_account_n
    umber():
       return ''.join(random.choice('0123456789') for _ in range(8))

    def display_bank_info(self):
       print(f"Bank    Name:
       {self.name}")
       print("Customers:")
       for     customer     in
       self.customers:
          customer.display_customer
          _info() print()

    def find_account_by_number(self, account_number):
       for       customer       in
          self.customers:      for
          account                 in
          customer.accounts:
             if account.account_number == account_number:
                return account
       return None


#         Example
usage if __name
_____==        "
main__":
   # Create a bank
   my_bank         =
   Bank("My
   Bank")
   customer_list=[] while True: print("1. New Customer 2. Existing
   Customer 3. Find Customers info 4.Exit") try:
         choice = int(input())

         if choice==1:
            print("Customer Registration: \n")
            #     Create     a     customer
            name=input("Enter  Customer
            Name:") address=input('Enter
            Customer Address: ')
            contact_number=input("Enter  Customer  Contact  Number:  ")  customer_obj  =
            Customer(name,  address,  contact_number)  customer_list.append(customer_obj)
            my_bank.add_customer(customer_obj) while True: acc_type = int(input("Enter 1. To
            create Saving account 2. To Create Cheking account 3. Exit\n")) if acc_type == 1:
```

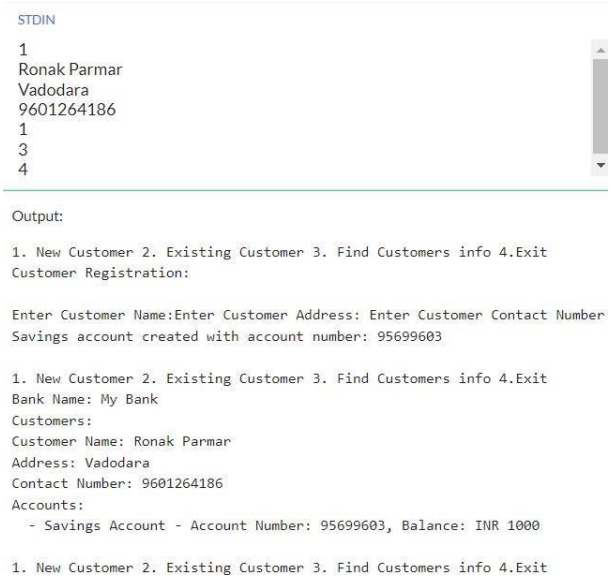
```
            new_account = customer_obj.create_account("Savings", 1000)
            print(f"Savings account created with account number:
            {new_account.account_number}\n") break
        elif acc_type == 2:
            new_account = customer_obj.create_account("Current", 1000)
            print(f"Current account created with account number:
            {new_account.account_number}\n") break

        elif acc_type == 3:
          break
        else:
          print("Invalid option. .. Try again")

if choice==2:
    # User input for transactions
    account_number_input = input("Enter your
    account number: ")
    account_to_transact = my_bank.find_account_by_number(account_number_input)

    if account_to_transact:
      print(f"\nWelcome,
      {account_to_transact.owner.name}!")
      print(account_to_transact) while True:
        print("1. Enter 1 to deposit\n2. Enter 2 to Withdrawl\n3. Enter 3 to Check the
        Balance\n4. Exit") option=int(input("Enter your Option:\n"))
        if option==1:
          print("Welcome to Deposit
          Section\n")
          # Deposit
          deposit_amount = int(input("\nEnter the amount to deposit: INR "))
          account_to_transact.deposit(deposit_amount)
        elif option==2: print("Welcome
          to withdrawl section:\n")
          # Withdrawal
          withdrawal_amount = int(input("\nEnter the amount to withdraw: INR "))
          account_to_transact.withdraw(withdrawal_amount)
        elif option==3:
          # Display updated account
          information
          print("\nUpdated Account
          Information:")
          print(account_to_transact)
        elif option==4:
          break
        else:
          print("Inv
```

```
                alid
                Option")
        else:
            print("Account not found.")
    if                  choice==3:
        my_bank.display_bank_info
        ()
    elif choice==4:
        break
    else:
        pass
except ValueError: print("Invalid input.
    Please enter a valid option.")
    continue
```

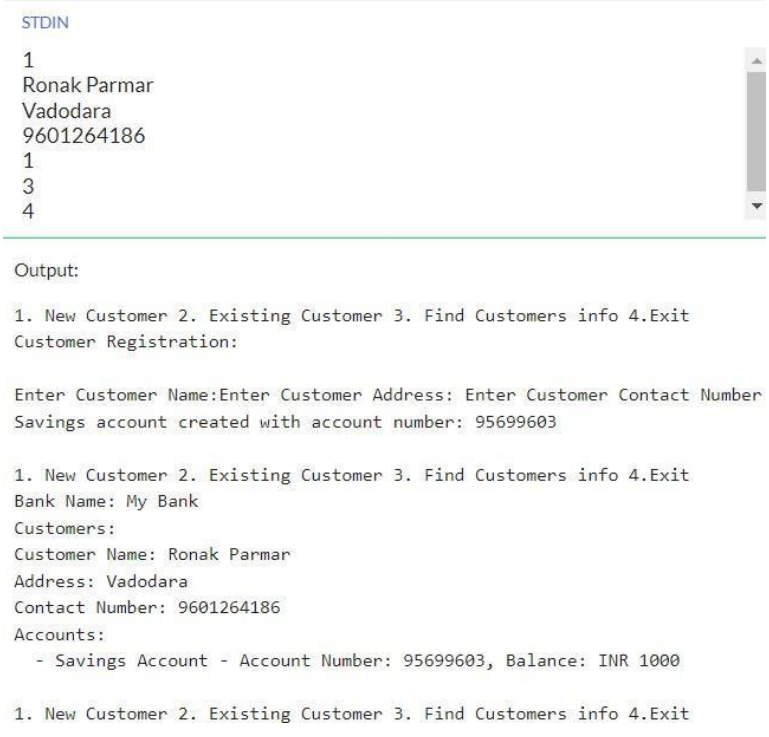
Class Diagram:



Expected Output:



Actual Output:



Result:

The result will be a program that allows users to create bank accounts, perform transactions, and manage customer and account information

PRACTICAL- 2

AIM: A program that simulates a school management system, with classes for the students, the teachers, and the courses.

Problem Statement:

Develop a program that simulates a school management system, modeling classes for students, teachers, and courses. The system should allow for the creation of students and teachers, enrollment in courses, grading, and tracking of academic information.

Program Description:

The program will use object-oriented programming to represent students, teachers, and courses. Students can enroll in courses, teachers can assign grades, and the system will maintain academic records for each student.

Algorithm:

1. Define a **Student** class with attributes like student ID, name, address, contact details, and a method to enroll in courses.
2. Create a **Teacher** class with attributes such as teacher ID, name, subject expertise, and a method to assign grades to students.
3. Implement a **Course** class with attributes like course code, course name, and a list of enrolled students.
4. Develop a **SchoolManagementSystem** class to manage students, teachers, and courses, allowing the creation of new students and teachers, enrollment in courses, and grading.

Source Code:

```
class Student:
    """
    Represents a student with ID, name, and grade.
    """

    def __init__(self, student_id, name, grade):
        self.student_id = student_id
        self.name = name
```

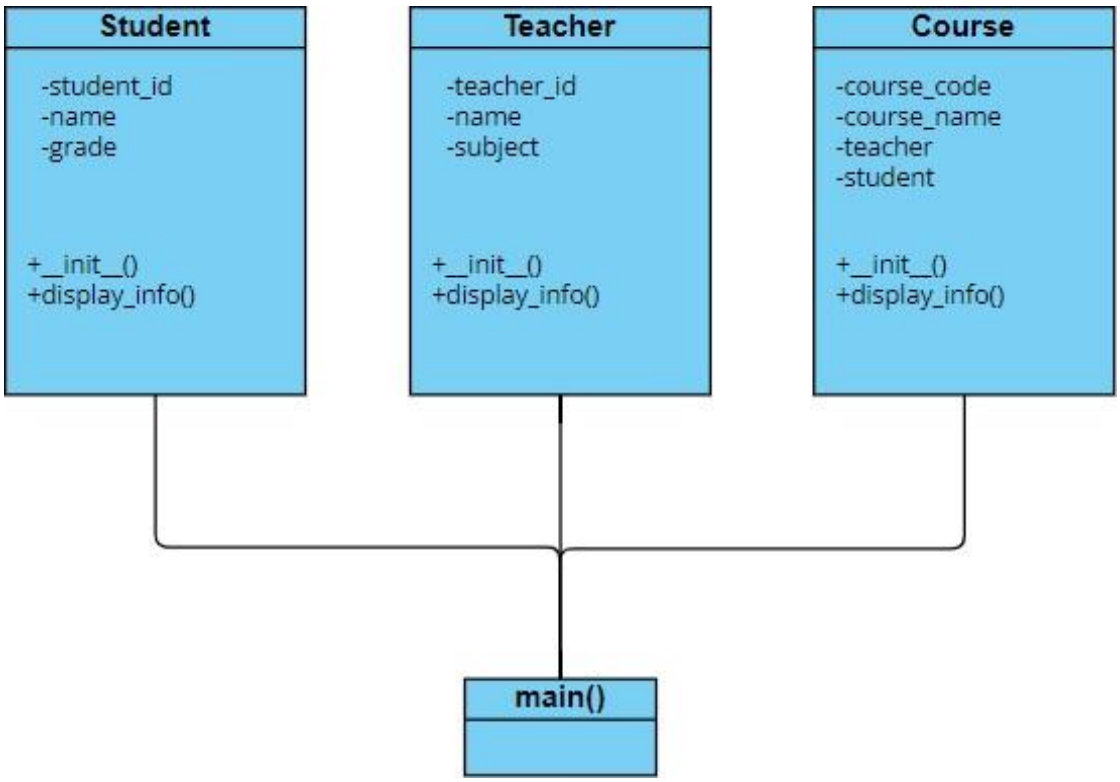
```
        self.grade = grade

    def display_info(self):
        """
        Prints the student's details in a formatted way.
        """
        print(f"\nStudent ID: {self.student_id}, Name: {self.name}, Grade: {self.grade}")


class Teacher:
    """
    Represents a teacher with ID, name, and subject.
    """

    def __init__(self, teacher_id, name, subject):
        self.teacher_id = teacher_id
        self.name = name
        self.subject = subject

    def display_info(self):
        """
        Prints the teacher's details in a formatted way.
        """
        print(f"\nTeacher ID: {self.teacher_id}, Name: {self.name}, Subject: {self.subject}")


class Course:
    """
    Represents a course with code, name, teacher, and enrolled students.
    """

    def __init__(self, course_code, course_name, teacher, students):
        self.course_code = course_code
        self.course_name = course_name
        self.teacher = teacher
        self.students = students

    def display_info(self):
        """
        Prints the course details, teacher information, and student details.
        """
        print(f"\nCourse Code: {self.course_code}, Course Name: {self.course_name}")
        print("\nTeacher:")
        self.teacher.display_info()
        print("\nStudents:")
```

```
        for student in self.students:
            student.display_info()


def main():
    """
    Main function to manage students, teachers, and courses.
    """
    students = []
    teachers = []
    courses = []

    while True:
        print("""
        1. Student Management
        2. Teacher Management
        3. Course Management
        4. Exit
        """)
        choice = int(input("\nEnter your choice: "))

        if choice == 1:
            # Student Management
            while True:
                print("""
                1. Add Student
                2. View Students
                3. Back
                """)
                student_choice = int(input("\nEnter your choice: "))

                if student_choice == 1:
                    # Add Student
                    student_id = input("\nEnter student ID: ")
                    name = input("\nEnter student name: ")
                    grade = input("\nEnter student grade: ")
                    students.append(Student(student_id, name, grade))
                    print("\nStudent added successfully.")
                elif student_choice == 2:
                    # View Students
                    if not students:
                        print("\nNo students registered yet.")
                    else:
                        for student in students:
                            student.display_info()
```

```
        elif student_choice == 3:
            # Back
            break
        else:
            print("\nInvalid input.")

    elif choice == 2:
        # Teacher Management
        while True:
            print("""
            1. Add Teacher
            2. View Teachers
            3. Back
            """)
            teacher_choice = int(input("\nEnter your choice: "))

            if teacher_choice == 1:
                # Add Teacher
                teacher_id = input("\nEnter teacher ID: ")
                name = input("\nEnter teacher name: ")
                subject = input("\nEnter teacher subject: ")
                teachers.append(Teacher(teacher_id, name, subject))
                print("\nTeacher added successfully.")
            elif teacher_choice == 2:
                # View Teachers
                if not teachers:
                    print("\nNo teachers registered yet.")
                else:
                    for teacher in teachers:
                        teacher.display_info()
            elif teacher_choice == 3:
                # Back
                break
            else:
                print("\nInvalid input.")

    elif choice == 3:
        # Course Management
        while True:
            print("""
            1. Add Course
            2. View Courses
            3. Back
            """)
            course_choice = int(input("\nEnter your choice: "))
```

```
if course_choice == 1:
    # Add Course
```

Class Diagram:



Expected Output:

STDIN

```
1
1
34090394
Ronak Parmar
80
```

Output:

```
1.Student_form/details
2.Teacher_form/details
3.Course_form/details

Enter your choice:
Enter the number of students:
Enter student 1 ID:
Enter student 1 name:
Enter student 1 grade:
Registration successful.
```

Actual Output:

STDIN

```
1
1
34090394
Ronak Parmar
80
```

Output:

```
1.Student_form/details
2.Teacher_form/details
3.Course_form/details

Enter your choice:
Enter the number of students:
Enter student 1 ID:
Enter student 1 name:
Enter student 1 grade:
Registration successful.
```

Result:

The program successfully calculates the area and perimeter of a rectangle based on the user's choice. It provides a user-friendly interface and accurate results for the calculations, meeting the goal of creating a rectangle calculator.

PRACTICAL- 3

AIM: A program that reads a text file and counts the number of words in it.

Problem Statement:

Develop a program that reads a text file and counts the number of words in it.

Program Description:

The program will take a text file as input, read its content, and count the number of words. A word is defined as any sequence of characters separated by whitespace.

Algorithm:

1. Open the text file in read mode.
2. Read the content of the file.
3. Tokenize the content based on whitespace to extract words.
4. Count the number of words.
5. Print the result.

Source Code:

```
    def count(path):
        try:
            with open(path,'r') as file:
                file_content     =     file.read()     return     f"data     =
                {file_content.split()}\nlength of the words:
    {len(file_content.split())}"
        except FileNotFoundError:
            return "Please Provide valid file path."
path ="example.txt"
    print(count(path))
```

Expected Output:

Output:

```
data = ['Parul', 'University,', 'Be', 'here,', 'Be', 'vibrant.']
length of the words: 6
```

Actual Output:

Output:

```
data = ['Parul', 'University,', 'Be', 'here,', 'Be', 'vibrant.']
length of the words: 6
```

Result:

The result will be the count of words in the specified text file.

PRACTICAL- 4

AIM: A program that reads a CSV file and calculates the average of the values in a specified column.

Problem Statement:

Develop a program that reads a CSV file, extracts data from a specified column, and calculates the average of the values in that column.

Program Description:

The program will take a CSV file as input, read its content, and allow the user to specify a column for which the average should be calculated. It will then perform the necessary calculations and display the average value.

Algorithm:

1. Accept the CSV file path and column name from the user.
2. Open and read the CSV file, extracting the specified column data.
3. Convert the column values to numerical format (assuming the values are numerical).
4. Calculate the average of the values in the specified column.
5. Display the result.

Source Code:

```
import csv


def calculate_average(csv_file, column_name):
    """
    Calculates the average value in a specific column of a CSV file.

    Args:
        csv_file (str): Path to the CSV file.
        column_name (str): Name of the column to calculate the average for.

    Returns:
        float: The average value in the specified column, or None if errors occur.
    """

    try:
```

```
        with open(csv_file, 'r') as file:
            reader = csv.DictReader(file)
            if column_name not in reader.fieldnames:
                print(f"Column '{column_name}' not found in the CSV file.")
                return None
            total = 0
            count = 0
            for row in reader:
                try:
                    value = float(row[column_name])
                    total += value
                    count += 1
                except ValueError:
                    print(f"Skipping row {reader.line_num}: Invalid value in column '{column_name}'.")
            if count == 0:
                print(f"No valid values found in column '{column_name}'.")
                return None
            average = total / count
            return average

    except FileNotFoundError:
        print(f"File '{csv_file}' not found.")
        return None


csv_file_path = 'file.csv'
column_to_calculate = 'ENGLISH'
result = calculate_average(csv_file_path, column_to_calculate)
if result is not None:
    print(f"The average value in column '{column_to_calculate}' is: {result}")
```

file.csv:

	A	B	C	D	E	F	G	H
1	Sr No.	Name	Enrollment	MATHS	CN	OS	PFSD	CC
2	1	Rakesh	2203051240086	80	4	44	80	22
3	2	Ritesh	2203051240112	99	45	77	70	55
4	3	Rohit	2203051240089	3	71	23	44	71
5	4	Rutal	2203051240124	33	44	23	70	3
6	4	Gautam	2203051240096	40	4	33	44	3
7	5	Pritesh	2203051240080	80	34	23	80	34
8	6	Raju	2203051249002	20	22	20	34	20
9	7	Ramesh	2203051240094	50	2	22	32	50
10	8	Sudeep	2203051240121	22	11	50	3	50
11	9	Sanjay	2203051240125	11	4	81	44	81
12	10	Jethalal	2203051240115	44	33	80	53	23

Expected Output:

```
Output:

Skipping row 14: Invalid value in column 'CN'.
The average value in column 'CN' is: 58.363636363333335
```

Actual Output:

```
Output:

Skipping row 14: Invalid value in column 'CN'.
The average value in column 'CN' is: 58.363636363333335
```

Result:

The result is the average value of the specified column in the provided CSV file. You can customize and expand this program based on your specific requirements and the structure of your CSV file.

PRACTICAL- 5

AIM: A program that reads an Excel file and prints the data in a tabular format.

Problem Statement:

Develop a program that reads an Excel file and prints its data in a tabular format.

Program Description:

The program will take an Excel file as input, read its content, and display the data in a tabular format. It may use libraries like **openpyxl** or **pandas** to handle Excel file operations.

Algorithm:

1. Accept the Excel file path from the user.
2. Open and read the Excel file.
3. Extract the data from the sheets.
4. Display the data in a tabular format.

Source Code:

```
import pandas as pd import

openpyxl output =

pd.read_excel("delimited.xlsx")

print(output)
```

delimited.xlsx:

	A	B	C	D	E	F	G	H
1	Sr No.	Name	Enrollment	MATHS	CN	OS	PFSD	CC
2	1	Rakesh	2203051240086	80	4	44	80	22
3	2	Ritesh	2203051240112	99	45	77	70	55
4	3	Rohit	2203051240089	3	71	23	44	71
5	4	Rutal	2203051240124	33	44	23	70	3
6	4	Gautam	2203051240096	40	4	33	44	3
7	5	Pritesh	2203051240080	80	34	23	80	34
8	6	Raju	2203051249002	20	22	20	34	20
9	7	Ramesh	2203051240094	50	2	22	32	50
10	8	Sudeep	2203051240121	22	11	50	3	50
11	9	Sanjay	2203051240125	11	4	81	44	81
12	10	Jethalal	2203051240115	44	33	80	53	23

Expected Output:

Output:

	Sr No.	Name	Enrollment	MATHS	CN	OS	PFSD	CC
0	1	Rakesh	2203051240086	80	4	44	80	22
1	2	Ritesh	2203051240112	99	45	77	70	55
2	3	Rohit	2203051240089	3	71	23	44	71
3	4	Rutal	2203051240124	33	44	23	70	3
4	4	Gautam	2203051240096	40	4	33	44	3
5	5	Pritesh	2203051240080	80	34	23	80	34
6	6	Raju	2203051249002	20	22	20	34	20
7	7	Ramesh	2203051240094	50	2	22	32	50
8	8	Sudeep	2203051240121	22	11	50	3	50
9	9	Sanjay	2203051240125	11	4	81	44	81
10	10	Jethalal	2203051240115	44	33	80	53	23

Actual Output:

Output:

	Sr No.	Name	Enrollment	MATHS	CN	OS	PFSD	CC
0	1	Rakesh	2203051240086	80	4	44	80	22
1	2	Ritesh	2203051240112	99	45	77	70	55
2	3	Rohit	2203051240089	3	71	23	44	71
3	4	Rutal	2203051240124	33	44	23	70	3
4	4	Gautam	2203051240096	40	4	33	44	3
5	5	Pritesh	2203051240080	80	34	23	80	34
6	6	Raju	2203051249002	20	22	20	34	20
7	7	Ramesh	2203051240094	50	2	22	32	50
8	8	Sudeep	2203051240121	22	11	50	3	50
9	9	Sanjay	2203051240125	11	4	81	44	81
10	10	Jethalal	2203051240115	44	33	80	53	23

Result:

The result is the data from the Excel file displayed in a structured tabular format. You can customize and extend this program based on your specific requirements and the structure of your Excel file.