Question 1 (30 marks)

Three project groups have been setup. Each group has at least one person. One of them is the group leader.

The member list of each group is stored in a text file. In each file, the first line contains the count of members in the group; followed by the list of member names, where the first one is the leader.

Please read the given **main()** method of a program, as well as the contents of 3 input files to set up the groups. The groups are named *Ace Team*, *Best Gang*, and *The Crown*.

You are to complete all required classes (**Group.java** and **Person.java**) for the program to set up the groups from the file contents and provides group listing, member searching, carries out the change (if valid) when *one member wants to change from his existing group to a new group*, and shows the history of a person.

The program should give the outputs as shown in the comments within the square brackets [] inside **main()**.

Remarks:

- Assume that the names of the people are all different. That is, no two persons have the same name.
- Assume that each person can join only one group at a time.
- After all operations, the updated list would be written to output files. We omit the details here. That is, this part is not needed to be implemented in the program.

Question 2 (20 marks)

Complete the given program by providing all required code inside the given Staff class.
Requirements:
- The program allocates staff members to handle jobs.
- Each job is to be worked by one staff member.
- Each staff can work for at most 1 job at any moment.
- Each staff has 1 or zero helper.
- All staff are lazy. Upon receiving a job request, a staff tries to tell his helper (if any) to handle the job. The staff himself (if free) will do it only if his helper cannot handle the job. (Of course his helper has the same way of handling the job request.)

Please read the given **main()** method of the program, as well as the given classes.

The program should give the outputs as shown in the comments within the square brackets [] inside **main()**.

Question 3 (22 marks)

You are to complete a program that creates **Boolean expressions** and evaluates them. A Boolean expression can be a *constant*, a *variable*, an *AND*, *OR*, or *NOT* operation.

Your task: Please read the given **main()** method of the program. Write all necessary classes/interfaces.

The program should give the outputs as shown in the comments within the square brackets [] inside **main()**.

Question 4 (28 marks)

You are to complete the program of an admission system that handles students' applications to different programmes.

Explanation and requirements:

- A student can submit one or more applications, one for each programme that he/she would like to study.
- When an application record is created, the initial status is set as ***Pending***.
- When a programme gives an offer to an applicant, the applications' status will be changed to ***Offered***.
- If the student accepts the offer, the application's status will be changed to ***Accepted***.
- Each student can enquire about the status of all his/her applications.
- The system can generate a report for each programme, showing the status of each application.

Please read the given **main()** method of the program, as well as the given classes. The program should give the outputs as shown in the comments within the square brackets [] inside **main()**.

Your task:

Study the given Student and Programme classes . According to main(), some methods are missing (namely accept, enquire, offer, and generateReport). They will be completed by you.

Moreover, you will implement the Application class, as well as additional modules (**IApply** and **IAdmit**) required by **Student** and **Programme**. Note that **Student** and **Programme** do not depend on **Application** (to conform to the *Dependency Inversion Principle*); they should depend on **IApply** and **IAdmit**.

You also need to model the status of programme applications using the *State Pattern*.

Your work should include the following:

- Finish the given **Student** and **Programme** classes
  Note: DO NOT add any field in these two classes; DO NOT *delete* any given code.
- Implement the **Application** class
- Implement all other required modules