You’ll have to create an application to manage Accounts. This application allows you register 2 types of account: Admin and User.

**1. interface named IActions<T>: (1p)**

* void Add(T acc);
* void Edit(T acc);
* void Delete(string userName);
* void View();
* T GetByUsername(string userName);
* bool Login(string userName, string Password);
* void Register();

**2. Create a class named Account: (1p)**

* Properties:
  + String UserName
  + String Password
  + String PhoneNumber
* 2 constructors:
  + Account(): Ask user to input UserName, Password and PhoneNumber and set them to properties.
  + Account(string Username, string Password, string PhoneNumber): set these parameters to properties.

3. Create a class named **Admin** inherit from **Account**: **(1,5p)**

* Properties named Role, Department (string).
* Implement both constructor from account:
  + public Admin():base() – Ask user to input Role and Department
  + public Admin(string Username, string Password, string PhoneNumber, string Role, string Department):base(Username,Password,PhoneNumber) – set Role and Department parameters to properties
* Role only accept either string “Admin” or “SAdmin”
* Department only accept either of strings: “HR”, “Marketing”, “Code”, “Design”

4. Create a class named **User** inherit from **Account**: **(1,5p)**

* Properties:
  + string Email
  + DateTime DOB
* Implement both constructor from account:
  + public User():base() – Ask user to input Email and DOB
  + public User(string Username, string Password, string PhoneNumber, string Email, DateTime DOB) :base(Username,Password,PhoneNumber)– set Email and DOB parameters to properties
* Use regular expression to ensure Email is input correctly
* DOB must be input in format: dd/MM/yyyy

5. Create a Class named UserManagement: **(5,5p)**

1. Implements the IAction (0,5p)
2. Create a generic collection object to store all the User. (0,5p)
3. Register will create new object User and call to Add function to add user to collection. (0,5p)
4. Login will check whether user information to login correct. (1p)
5. Edit method allows you change the user information by: Search from the list for user with the username equal to UserName being entered, if found then allow input new information except Password for the user (PhoneNumber, Email, DOB) (1p)
6. Delete method allows you remove a user from the list.  (0,5p)
7. View method will show all user in the list. (0,5p)
8. GetByUsername method will return an user by given username, if there’s no user with that given username, return null. (1p)

6. Create a Class named AdminManagement: **(3p)**

1. Implements the IAction (0,5p)
2. Create a generic collection object to store all the Admin. (0,5p)
3. Register will create new object User and call to Add function to add admin to collection. (0,5p)
4. Login will check whether admin UserName and Password to login correct. (0,5p)
5. All other methods (View, GetByUserName, Delete and Edit) will throw UnauthorizedAccessException exception. (1p)

7. In Program class: **(6,5p)**

Display a task menu to choose

Create an instance of UserManagement and AdminManagement.

Menu:

1. Login as Admin
2. Login as User
3. Register as new Admin
4. Register as new User
5. Exit

a. If user type 1 from keyboard: **(0,5p)**

Ask user to enter UserName and Password, then check it against Admin collection. If Login correct, go to step l else go back to previous menu.

b. If user type 2 from keyboard: **(0,5p)**

Ask user to enter UserName and Password, then check it against User collection. If Login correct, go to step m else go back to previous menu.

c. If user type 3 from keyboard: **(0,5p)**

Call to Register function of Admin object

d. If user type 4 from keyboard: **(0,5p)**

Call to Register function of User object.

e. If user type 5 from keyboard: **(0,5p)**

Exit the application.

f. While user not chooses either of options (1-5) then go back to the menu to ask user chooses on option. **(0,5p)**

g. If user login correctly as Admin, show a new Menu:

1. Add new User

2. Edit an User

3. Delete an User

4. Show all User

5. Go back

h. If user type 1: **(0,5p)**

Ask user to enter a new User information then add the new User to collection using Add method.

i. If user type 2: **(0,5p)**

Ask user to enter Username to edit, if there is User exist in collection with Username input, ask the user to input new User’s information to edit (PhoneNumber, Email, DOB) then use Edit method to edit the User. If User with Username doesn’t exist, show “User not found” message then go back to menu.

j. If user type 3: **(0,5p)**

Ask user to enter Username to delete, if found, Delete User from collection using Delete method. If not, show “User not found” message then go back to menu.

k. If user type 4: **(0,5p)**

Show all the User in collection to console, each user in 1 line.

l. If user type 5: **(0,5p)**

Go back to previous menu.

m. If user login correctly as User: **(1p)**

Show a welcome message: “Welcome “+Username.

If Username length > 2, Show a tree under the welcome message with height equal to the length of Username as following examples:

If Username = Thang, then length of Username = 5, show following message and tree:

Welcome Thang

\*

\*\*\*

\*\*\*\*\*

\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*

\*

If Username = Huy then length of Username = 3, show following message and tree:

Welcome Huy

\*

\*\*\*

\*\*\*\*\*

\*

If Username = Ha, then the length of Username = 2, show only the message.

Welcome Ha