The goal of this mini-project is to create easy-to-use shell scripts that allow someone who is not expert to adds users or remove them on a Linux system.

You should work on the project **individually**, at home or during the help sessions. You can ask the teaching assistants for help if you have any questions. When you finish, please upload the shell scripts (as .sh files) to Blackboard before the deadline. You can specify your name as a comment (starting with #) at the beginning of the source code (second line).

NOTE: Please be aware that any tentative of plagiarism/copying/modifying the source code from another student will be detected and will result in the project not being accepted from both students.

Part 1 - `create-user.sh`

Imagine that you're working as a Linux system administrator for a company. You are very busy, but you are constantly being interrupted by the helpdesk calling you to create new Linux accounts for all the people newly recruited by the company. You decide to write an easy-to-use shell script that will create new user accounts. Once you're done with the shell script you can put the helpdesk in charge of creating new accounts which will finally allow you to work uninterrupted.

You think about what the shell script must do and how you would like it to operate. You come up with the following list. The script:

- Enforces that it be executed with superuser (root) privileges. If the script is not executed with superuser privileges it will not attempt to create a user and returns an exit status of 1.
  Help: if the script is executed with root privileges (i.e. with `sudo`) as expected, then the environment variable UID should contain 0.
- Prompts the person who executed the script to enter the username (login), the full name for the person who will be using the account, and the initial password for the account.
- Creates a new user on the local system with the input provided by the user. Note: there should be a home directory created for the user.
- Informs the user if the account was not able to be created for some reason. If the account is not created, the script should return an exit status of 1.
- Sets the password (previously entered) for the created user. To do this, you can send the text `"username:password"` (i.e. the username and password entered previously) as an input to the command `chpasswd`. Think about using pipes for this.
- Force the user to change her/his password when she/he first login. Check the `man` page of the `passwd` command and search for a specific option to do this.
- Displays the username, password, and host where the account was created. This way the helpdesk staff can copy the output of the script in order to easily deliver the information to the new account holder.

Please give the name `create-user.sh` to your script. You can use the initial file provided to you with comments (check `create-user.sh`) to help you with the logic and flow of your script. Remember that the first time you execute the script you'll need to make sure it has executable permissions: `chmod 755 create-user.sh`

Here is an example run of the script. (Portions typed are in bold.)

```
sudo ./create-user.sh
Enter the username to create: satoshi
```

```
Enter the full name of the person account: Satoshi Nakamoto
Enter the password to use for the account: Ek3ABstract$
passwd: password expiry information changed.

Summary of the information:
Username: satoshi
Password: Ek3ABstract$
Host: ITE14563
```

Make sure the accounts have been created by examining the last 4 lines of the `/etc/passwd` file, and that a home directory has been created for that user.

```
tail -n 4 /etc/passwd
ls /home/
```

You can switch to the `satoshi` user to test if the script correctly forces a password change upon first login.

```
su – satoshi
…
…
exit
```

Also, test to make sure that the script exits with a non-zero exit status if the user does not use superuser (root) privileges. (Portions typed are in bold.)

```
./create-user.sh
Please run with sudo or as root.

echo ${?}
1
```

Part 2 - `create-user-improved.sh`

The helpdesk team has been using the `create-user.sh` script that you created. They're happy that they don't have to wait for you to create new accounts. However, they would like you to improve the script a little bit. They're tired of coming up with a unique password for each user they create. They think it would be great if the script automatically generated a password for each new account. Also, they think it would be more efficient if they could just specify the account username and account comments (i.e. full name) on the command line instead of being prompted for them.

Create a copy of your previous script `create-user.sh` and name it as `create-user-improved.sh`. Then, modify the script so that it:

- Enforces that it be executed with superuser (root) privileges. If the script is not executed with superuser privileges it will not attempt to create a user and returns an exit status of 1.
- Provides a usage statement (help message) if the user runs the script without providing any arguments (i.e. if the user does not supply an account username on the command line) and returns an exit status of 1.
- Uses the first argument provided on the command line as the username for the account. Any remaining arguments on the command line will be treated as the comment for the account.

- Automatically generates a password for the new account. Note: as previously, the user is expected to be later forced to change her/his password upon the first login.
- Informs the user if the account was not able to be created for some reason. If the account is not created, the script is to return an exit status of 1.
- Displays the username, password, and host where the account was created.

Note also, that helpdesk want the improved script to only display the details that they need to send to the user after they create their account, without displaying additional messages such as: `password expiry information changed. etc.`

Some help: one way to generate a sequence of (e.g. 10) random characters is as follows. You can use the command `date +%s%N` to get the current date in the form of a number of seconds and current nanoseconds. Send (using pipes) this number as an input to the command `sha256sum` in order to compute its hash. Send this hash as an input to the command `head -c10` in order to take the first 10 characters.

Don't forget to test your improved script the same way as previously to see if it works as expected.

Part 3 - `delete-users.sh`

Today the helpdesk team received a request to delete an account for a person who has left the company. The helpdesk team also wants to save the contents of this person's home directory just in case this person wants to return. They realize this is just going to be the first of these types of requests. They know they're eventually going to receive requests to remove accounts for people who leave and change department. Of course, you know it's going to be way easier to write a script and hand it off than it is for you to do all the work.

You think about what the shell script must do and how you would like it to operate. You come up with the following list. The script:

- Enforces that it be executed with superuser (root) privileges. If the script is not executed with superuser privileges it will not attempt to delete a user and returns an exit status of 1.
- Checks if a directory `/archive` exists and creates it if not. This directory will serve to save compressed version of the home directories for the deleted users.
- Accepts a list of usernames (to be deleted) as arguments. At least one username is required, or the script will display a usage and return an exit status of 1.
- Only allow the helpdesk team to change user accounts (not system accounts). So, the script should refuse to delete any accounts that have a UID less than 1000.
- Informs the user if an account was not able to be deleted, or if the corresponding home directory was not able to be compressed and saved for some reason.
- Displays the username and any actions performed for the account.

Please give the name `delete-users.sh` to your script. You can use the initial file provided to you with comments (check `delete-users.sh`) to help you with the logic and flow of your script.

Once you are done, test the script by:

- Executing it without root privileges.
- Executing it with root privileges, but without any arguments.
- Attempting to disable a system account (instead of a user account).
- Deleting the one user account (that you should have created beforehand).
- Deleting several user accounts (that you should have created beforehand).
- Also, check if the home directories have been removed, compressed, and saved correctly.