Lab9

Step1:

```
┌──(sahmer㉿kali-270)-[~]
└─$ sudo useradd Alice
[sudo] password for sahmer:
useradd: user 'Alice' already exists

┌──(sahmer㉿kali-270)-[~]
└─$
```

Step2:

```
  GNU nano 7.2                                                1ab9.sh
#!/bin/bash
echo "please enter your midas:"
read midas
current_date=$(date "+%Y.%m.%d-%H.%M.%S")
echo" time now :$current_date "
#tar file= midas id- current date.tar
tar_file='/var/backups/'$midas-$current_date'.tar'
tar -cvf $tar_file /home/Alice/
gzip $tar_file
```

Step3:

```
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
 */3 * * * * /home/sahmer/1ab9.sh
#
```

```
└─$ ls /var/backups
-2023.11.06-17.04.01.tar.gz  apt.extended_states.0  dpkg.diversions.0    dpkg.status.0
alternatives.tar.0           dpkg.arch.0           dpkg.statoverride.0  sisma002-2023.11.01-16.16.01.gz

┌──(sahmer㉿kali-270)-[~]
└─$ ls /var/backups
-2023.11.06-17.04.01.tar.gz  alternatives.tar.0     dpkg.diversions.0    sisma002-2023.11.01-16.16.01.gz
-2023.11.06-17.06.01.tar.gz  apt.extended_states.0  dpkg.statoverride.0
-2023.11.06-17.09.01.tar.gz  dpkg.arch.0           dpkg.status.0
```

Step4:

```
┌──(sahmer㉿kali-270)-[~]
└─$ sudo crontab -r

┌──(sahmer㉿kali-270)-[~]
└─$ sudo crontab -l
no crontab for root
```

Task B: Extra credit

```bash
#!/bin/bash

# Threshold will equal the number of backups
Threshold=3

# Get the number of backup archives in /var/backups/
number_backups=$(ls -l /var/backups/*.tar.gz 2>/dev/null | wc -l)

# Checking to make sure the amount of backups is greater than Threshold
if [ "$number_backups" -gt "$Threshold" ]; then
    # Calculate backups deleted
    backups_deleted=$((number_backups - Threshold))

    # Old backups archives to delete
    ls -t /var/backups/*.tar.gz | tail -n "$backups_deleted" | xargs rm
    echo "Deleted $backups_deleted old backups."
else
    echo "There are no current old backups to display."
fi
```

```
└─$ ./cleaner.sh
rm: cannot remove '/var/backups/-2023.11.06-17.21.01.tar.gz': Permission denied
rm: cannot remove '/var/backups/-2023.11.06-17.18.01.tar.gz': Permission denied
rm: cannot remove '/var/backups/-2023.11.06-17.15.01.tar.gz': Permission denied
rm: cannot remove '/var/backups/-2023.11.06-17.12.01.tar.gz': Permission denied
rm: cannot remove '/var/backups/-2023.11.06-17.09.01.tar.gz': Permission denied
rm: cannot remove '/var/backups/-2023.11.06-17.06.01.tar.gz': Permission denied
rm: cannot remove '/var/backups/-2023.11.06-17.04.01.tar.gz': Permission denied
Deleted 7 old backups.
```