Name: *Krithika Swamninathan*
Reg. No.: 205001057

Assignment 8 – Control flow, Loops, Functions and Arrays

Exercise 1 (Conditional statements)

1. If cost price and selling price of an item is input through the keyboard, write a program to determine whether the seller has made profit or incurred loss. Also determine how much profit was made or loss incurred.

```
asec20@sel20-HP-Compaq-Pro-6305-SFF:~/krith$ cat sales.sh
read -p "Enter cost price: " cp
read -p "Enter selling price: " sp
profit=$(($sp-$cp))
if [ $profit -lt 0 ]
        then
        echo "Loss is $((-$profit))."
elif [ $profit -gt 0 ]
        then
        echo "Profit is $profit."
else
        echo "No profit or loss!"
fi

asec20@sel20-HP-Compaq-Pro-6305-SFF:~/krith$ bash ./sales.sh
Enter cost price: 40
Enter selling price: 50
Profit is 10.

asec20@sel20-HP-Compaq-Pro-6305-SFF:~/krith$ bash ./sales.sh
Enter cost price: 120
Enter selling price: 100
Loss is 20.

asec20@sel20-HP-Compaq-Pro-6305-SFF:~/krith$ bash ./sales.sh
Enter cost price: 50
Enter selling price: 50
No profit or loss!
```

2. Write a shell script to validate password strength. Here are a few assumptions for the password string.
•Length – minimum of 8 characters.
•Contain both alphabet and number.
•Include both the small and capital case letters.
If the password doesn't satisfy with any of the above conditions, then the script should print it as a "Weak Password"

```
asec20@sel20-HP-Compaq-Pro-6305-SFF:~/krith$ cat password.sh
read -sp "Password: " pd
len=${#pd}
if [ $len -ge 8 ]
```

```
        then
        if [[ $pd =~ [a-zA-Z0-9]* ]]
                then
                if [[ $pd =~ [a-zA-Z]+ && $pd =~ [A-Z]+ ]]
                        then
                        echo "strong password"
                else
                        echo "weak password"
        else
                echo "weak password"
        fi
else
        echo "weak password"
fi
^C

asec20@sel20-HP-Compaq-Pro-6305-SFF:~/krith$ bash password.sh
asec20@sel20-HP-Compaq-Pro-6305-SFF:~/krith$ bash ./a.sh
Enter password: 12345678
weak password

asec20@sel20-HP-Compaq-Pro-6305-SFF:~/krith$ bash ./a.sh
Enter password: a123b573
weak password

asec20@sel20-HP-Compaq-Pro-6305-SFF:~/krith$ bash ./a.sh
Enter password: AB123567
weak password

asec20@sel20-HP-Compaq-Pro-6305-SFF:~/krith$ bash ./a.sh
Enter password: wejrwoeijrower
weak password

asec20@sel20-HP-Compaq-Pro-6305-SFF:~/krith$ bash ./a.sh
Enter password: Password1234
strong password
```

3. Write a script that prints essentially the same information as ls -l a but in a more user- friendly way.
(a) file exists or not
(b) regular file?
(c) directory?
(d) readable?
(e) writable?
(f) executable?
(g) owner
Print suitable messages.
Rewrite the above script as a shell function finfo and call the function with a filename.

```
asec20@sel20-HP-Compaq-Pro-6305-SFF:~/krith$ cat file.sh
read -p "Enter file name: " file
if [[ -e $file ]]
then
        ls -ld $file | cut -d' ' -f3
        echo "Path exists"
        if [[ -d $file ]]
        then
                echo "Directory"
        elif [[ -f $file ]]
        then
                echo "Regular File"
                if [[ -r $file ]]
                then
                        echo "Readable file"
                elif [[ -w $file ]]
                then
                        echo "Writable file"
                elif [[ -x $file ]]
                then
                        echo "Executable file"
                else
                        echo "No permissions available"
                fi
        else
                echo "Not regular file"
        fi
else
        echo "Path does not exist"
fi

asec20@sel20-HP-Compaq-Pro-6305-SFF:~/krith$ bash ./file.sh
Enter file name: file.sh
asec20
Path exists
Regular File
Readable file
```

Exercise 2 (Loops)

1. Write a program to generate all combinations of digits 1, 2 and 3 to form different numbers using for loops.

```
asec20@sel20-HP-Compaq-Pro-6305-SFF:~/krith$ cat comb.sh
for i in {1..3}
do
        for j in {1..3}
        do
```

```
                for k in {1..3}
                do
                        echo "$i $j $k"
                done
        done
done
```

```
asec20@sel20-HP-Compaq-Pro-6305-SFF:~/krith$ bash ./comb.sh
1 1 1
1 1 2
1 1 3
1 2 1
1 2 2
1 2 3
1 3 1
1 3 2
1 3 3
2 1 1
2 1 2
2 1 3
2 2 1
2 2 2
2 2 3
2 3 1
2 3 2
2 3 3
3 1 1
3 1 2
3 1 3
3 2 1
3 2 2
3 2 3
3 3 1
3 3 2
3 3 3
```

2. Use seq with for statement to print the multiplication table.

```
asec20@sel20-HP-Compaq-Pro-6305-SFF:~/krith$ cat mul.sh
read -p "Enter number: " num
end=$(($num*10))
echo "Multiplication Table"
seq $num $num $end
```

```
asec20@sel20-HP-Compaq-Pro-6305-SFF:~/krith$ bash ./mul.sh
Enter number: 6
Multiplication Table
6
12
```

```
18
24
30
36
42
48
54
60
```

3. Write a shell script to check whether a given string is a palindrome or not

```
kri@kri-ubuntu:~/workspace/uni8$ cat pal.sh
read -p "Enter string to check for palindrome: " data
reverse=`echo "$data"|rev`
if [ "$data" == "$reverse" ]
then
        echo "$data is a palindrome"
else
        echo "$data is not a palindrome"
fi

kri@kri-ubuntu:~/workspace/uni8$ bash ./pal.sh
Enter string to check for palindrome: racecar
racecar is a palindrome

kri@kri-ubuntu:~/workspace/uni8$ bash ./pal.sh
Enter string to check for palindrome: home
home is not a palindrome
```

4. Write a shell script to compute ‘m’ to the power of a positive integer ‘n’, i.e. m n (while loop).

```
kri@kri-ubuntu:~/workspace/uni8$ cat power.sh
read -p "Enter m: " m
read -p "Enter n: " n
result=1
while (( n>0 ))
do
        result=$((result*m))
        (( n-- ))
done
echo "Result: $result"

kri@kri-ubuntu:~/workspace/uni8$ bash ./power.sh
Enter m: 2
Enter n: 5
Result: 32

kri@kri-ubuntu:~/workspace/uni8$ bash ./power.sh
```

```
Enter m: 7
Enter n: 3
Result: 343
```

5. Write a script that attempts to copy a file to a directory and, if it fails, waits 5 seconds and then tries again continuing until it succeeds. (Use Until statement).

```
kri@kri-ubuntu:~/workspace/uni8$ cat copy.sh
read -p "Enter file to copy: " file
read -p "Enter directory to copy to: " directory
until cp $file $directory/
do
echo "Trying to copy file to directory"
sleep 5
read -p "Enter file to copy: " file
read -p "Enter directory to copy to: " directory
done

kri@kri-ubuntu:~/workspace/uni8$ bash ./copy.sh
Enter file to copy: new.txt
Enter directory to copy to: folder
cp: cannot stat 'new.txt': No such file or directory
Trying to copy file to directory
Enter file to copy: mul.sh
Enter directory to copy to: folder
```

6. Write a menu based program to copy a given file, to remove the specified file and to move a file.

```
kri@kri-ubuntu:~/workspace/uni8$ cat menu.sh
echo "Menu"
echo "1. Copy a File"
echo "2. Remove a file "
echo "3. Move a file"
echo "4. Quit"
read -p "Enter your choice: " choice
case "$choice" in
        1) read -p "Enter file name to copy: " f1
        read -p "Enter file to copy to: " f2
        if [ -f $f1 ]
        then
                cp $f1 $f2
                echo "Original file: "
                cat $f1
                echo "Copied file: "
                cat $f2
        else
                echo "$f1 does not exist"
        fi
```

```
        ;;
2) read -p "Enter file to be removed: " r
        if [ -f $r ]
        then
                rm -i $r
        else
                echo "file $r does not exist"
        fi
        ;;
3) read -p "Enter source file: " f1
        read -p "Enter destination directory: " f2
        if [ -f $f1 ]
        then
                if [ -d $f2 ]
                then
                mv $f1 $f2
                else
                        echo "$f2 is not a directory"
                fi
        else
                echo "$f1 does not exist"
        fi
        ;;
4) echo "Exiting the program"
        exit;;
esac

kri@kri-ubuntu:~/workspace/uni8$ bash ./menu.sh
Menu
1. Copy a File
2. Remove a file
3. Move a file
4. Quit
Enter your choice: 1
Enter file name to copy: copy.sh
Enter file to copy to: folder
Original file:
read -p "Enter file to copy: " file
read -p "Enter directory to copy to: " directory
until cp $file $directory/
do
echo "Trying to copy file to directory"
sleep 5
read -p "Enter file to copy: " file
read -p "Enter directory to copy to: " directory
done
Copied file:
cat: folder: Is a directory
```

```
kri@kri-ubuntu:~/workspace/uni8$ bash ./menu.sh
Menu
1. Copy a File
2. Remove a file
3. Move a file
4. Quit
Enter your choice: 2
Enter file to be removed: new.txt
file new.txt does not exist
```

Exercise 3 (Function)

1.Write shell script to read a text file name and count the number of lines using function. Pass the file name as an argument to the function. Return the number of lines and print it.

```
kri@kri-ubuntu:~/workspace/uni8$ cat count.sh
lines(){
if [ -f "$1" ]
then
num=`wc -l $1 | cut -d" " -f1`
echo "File $1 has $num lines"
else
echo "$1 is not a file"
fi
}
lines $1

kri@kri-ubuntu:~/workspace/uni8$ bash ./count.sh
 is not a file

kri@kri-ubuntu:~/workspace/uni8$ cat> newfile
hi
bye
this is fun
no not fun
time to sleep
^C

kri@kri-ubuntu:~/workspace/uni8$ bash ./count.sh newfile
File newfile has 5 lines
```

2. Write a shell script to count the number of occurrences of given word in the file. (Note: File name and word to be passed as an argument to the script).

```
kri@kri-ubuntu:~/workspace/uni8$ cat occur.sh
occurence(){
file=$1
word=$2
```

```
if [ -f "$file" ]
then
num=`grep -w "$word" "$file" | wc -l| cut -d" " -f1`
echo "There are $num occurences of the word $word in $file."
else
echo "$1 is not a file"
fi
}
occurence $1 $2
```

```
kri@kri-ubuntu:~/workspace/uni8$ cat newfile
hi
bye
this is fun
no not fun
time to sleep

kri@kri-ubuntu:~/workspace/uni8$ bash ./occur.sh newfile fun
There are 2 occurences of the word fun in newfile.
```

3. Anna University converts the marks in an exam to letter grades according to the following table. Write a shell script to translate the marks of a student in a semester into letter grades.

Mark range	Grade points	Letter grade
91-100	**10**	**S**
81-90	**9**	**A**
71-80	**8**	**B**
61-70	**7**	**C**
57-60	**6**	**D**
51-56	**5**	**E**
< 50	**0**	**U**

```
kri@kri-ubuntu:~/workspace/uni8$ cat marks.sh
read -p "Enter marks: " mark
if (( mark >= 91 && mark<=100 ))
then
        echo "Grade point: 10 Letter grade: S"
elif (( mark>=81 && mark<=90 ))
then
        echo "Grade point: 9 Letter grade: A"
elif (( mark>=71 && mark<=80 ))
then
        echo "Grade point: 8 Letter grade: B"
elif (( mark>=61 && mark<=70 ))
then
        echo "Grade point: 7 Letter grade: C"
elif (( mark>=57 && mark<=60 ))
then
        echo "Grade point: 6 Letter grade: D"
```

```
elif (( mark>=51 && mark<=57 ))
then
        echo "Grade point: 5 Letter grade: E"
else
        echo "Grade point: 0 Letter grade: U"
fi
```

```
kri@kri-ubuntu:~/workspace/uni8$ bash ./marks.sh
Enter marks: 94
Grade point: 10 Letter grade: S

kri@kri-ubuntu:~/workspace/uni8$ bash ./marks.sh
Enter marks: 38
Grade point: 0 Letter grade: U

kri@kri-ubuntu:~/workspace/uni8$ bash ./marks.sh
Enter marks: 72
Grade point: 8 Letter grade: B
```

Exercise 4

1. Write a shell script that prints 5 command line arguments. What happens if we pass fewer than 5 arguments?

```
kri@kri-ubuntu:~/workspace/uni8$ cat print.sh
j=1
for i in "$@"
do
        echo $i
        j=$(( $j+1 ))
        if [[ $j > 5 ]]
        then
                break
        fi
done
```

```
kri@kri-ubuntu:~/workspace/uni8$ bash ./print.sh

kri@kri-ubuntu:~/workspace/uni8$ bash ./print.sh one two 3 4 FIVE
one
two
3
4
FIVE
```

2. Change the value of a positional parameter. Did you succeed?

```
kri@kri-ubuntu:~/workspace/uni8$ cat change.sh
```

```
echo "The parameters are: "
for i in "$@"
do
        echo $i
done
echo "Changing argument 2 to unix"
$2="unix"
echo "The parameters after the change are: "
for i in "$@"
do
        echo $i
done
```

```
kri@kri-ubuntu:~/workspace/uni8$ bash ./change.sh
The parameters are:
Changing argument 2 to unix
./change.sh: line 7: =unix: command not found
The parameters after the change are:
```

```
kri@kri-ubuntu:~/workspace/uni8$ bash ./change.sh abc def ghi
The parameters are:
abc
def
ghi
Changing argument 2 to unix
./change.sh: line 7: def=unix: command not found
The parameters after the change are:
abc
def
ghi
```

Note: We cannot change the values of positional parameters.

Exercise 5

1. Develop an interactive script to maintain a database of employees. The database is in the format

employee_name	rate_per_hour	hours_worked
Beth	**4.00**	**0**
Dan	**3.75**	**0**
Kathy	**4.00**	**10**
Mark	**5.00**	**20**
Mary	**5.50**	**22**
Susie	**4.25**	**18**

The script should allow users to
1. List the records
2. Search for an employee
3. Modify the hours_worked of an employee whose existing hours_worked is equal to 0.

Name: *Krithika Swamninathan*
Reg. No.: 205001057

4. Delete an employee
5. Quit

```
kri@kri-ubuntu:~/workspace/uni8$ cat database.sh
while [ 1 ]
do
echo "Menu"
echo "1. List records"
echo "2. Search for an employee "
echo "3. Modify the hours worked for an employee whose existing hours worked is
0"
echo "4. Delete employee"
echo "5. Quit"
read -p "Enter your choice: " choice
case "$choice" in
1) sed -n 'p' database
;;
2) read -p "Enter name to search for: " name
grep "^$name" database
if [[ $? != 0 ]]
then
echo "Employee not Found"
fi
;;
3) read -p "Enter hours to substitute: " hours
sed -ie "s/ 0$/ $hours/" database
;;
4) read -p "Enter name to search for: " name
sed -ie "s/^$name.*//" database
;;
5) break
;;
esac
done
```

```
kri@kri-ubuntu:~/workspace/uni8$ bash ./database.sh
Menu
1. List records
2. Search for an employee
3. Modify the hours worked for an employee whose existing hours worked is 0
4. Delete employee
5. Quit
Enter your choice: 1
Beth    4.00    0
Dan     3.75    0
Kathy   4.00    10
Mark    5.00    20
Mary    5.50    22
Susie   4.25    18
```

Name: *Krithika Swamninathan*
Reg. No.: 205001057

```
Menu
1. List records
2. Search for an employee
3. Modify the hours worked for an employee whose existing hours worked is 0
4. Delete employee
5. Quit
Enter your choice: 2
Enter name to search for: Dan
Dan     3.75    0

Menu
1. List records
2. Search for an employee
3. Modify the hours worked for an employee whose existing hours worked is 0
4. Delete employee
5. Quit
Enter your choice: 2
Enter name to search for: Alice
Employee not Found

Menu
1. List records
2. Search for an employee
3. Modify the hours worked for an employee whose existing hours worked is 0
4. Delete employee
5. Quit
Enter your choice: 3
Enter hours to substitute: 6

Menu
1. List records
2. Search for an employee
3. Modify the hours worked for an employee whose existing hours worked is 0
4. Delete employee
5. Quit
Enter your choice: 1
Beth    4.00    6
Dan     3.75    6
Kathy   4.00    10
Mark    5.00    20
Mary    5.50    22
Susie   4.25    18

Menu
1. List records
2. Search for an employee
3. Modify the hours worked for an employee whose existing hours worked is 0
4. Delete employee
5. Quit
```

```
Enter your choice: 4
Enter name to delete: Mary

Menu
1. List records
2. Search for an employee
3. Modify the hours worked for an employee whose existing hours worked is 0
4. Delete employee
5. Quit
Enter your choice: 1
Beth    4.00    0
Dan     3.75    0
Kathy   4.00    10
Mark    5.00    20

Susie   4.25    18

Menu
1. List records
2. Search for an employee
3. Modify the hours worked for an employee whose existing hours worked is 0
4. Delete employee
5. Quit
Enter your choice: 5
```

2. Create an array by assignment of prices for five different fruits with fruit name as key and price as value.
a. Display the all the key.
b. Display the values.
c. Display the key value pair.d. Remove the third fruit.
e. Add one new fruit.
f. Calculate the total cost of all fruits and display the amount.
g. Delete the all items and display

```
kri@kri-ubuntu:~/workspace/uni8$ cat fruits.sh
declare -A fruits
for i in {1..5}
do
        read -p "Enter fruit name: " name
        read -p "Enter price: " price
        fruits[$name]=$price
done

echo "Keys: "
for key in "${!fruits[@]}"; do echo $key; done
echo

echo "Values: "
for val in "${fruits[@]}"; do echo $val; done
```

```
echo

echo "Key and Values: "
for key in "${!fruits[@]}"
do
        echo "Key: $key Value: ${fruits[$key]}"
done
echo

echo "Deleting the third element: "
count=0
for key in "${!fruits[@]}"
do
        (( count++ ))
        if (( count == 3 ))
        then
                unset fruits[$key]
        fi
done
echo

echo "Key and Values: "
for key in "${!fruits[@]}"
do
        echo "Key: $key Value: ${fruits[$key]}"
done
echo

echo "Adding a new element: "
read -p "Enter fruit name: " name
read -p "Enter price: " price
fruits[$name]=$price
echo

echo "Key and Values: "
for key in "${!fruits[@]}"
do
        echo "Key: $key Value: ${fruits[$key]}"
done
echo

echo "Total cost: "
total=0
for value in "${fruits[@]}"
do
        total=$(( $total+$value ))
done
echo "Total price: " $total
echo
```

```
echo "Deleting associative array..."
unset fruits
echo "${fruits[@]}"
```

```
kri@kri-ubuntu:~/workspace/uni8$ bash ./fruits.sh
Enter fruit name: banana
Enter price: 40
Enter fruit name: grapes
Enter price: 50
Enter fruit name: pineapple
Enter price: 80
Enter fruit name: apple
Enter price: 60
Enter fruit name: mango
Enter price: 30
Keys:
grapes
pineapple
mango
banana
apple

Values:
50
80
30
40
60

Key and Values:
Key: grapes Value: 50
Key: pineapple Value: 80
Key: mango Value: 30
Key: banana Value: 40
Key: apple Value: 60

Deleting the third element:

Key and Values:
Key: grapes Value: 50
Key: pineapple Value: 80
Key: banana Value: 40
Key: apple Value: 60

Adding a new element:
Enter fruit name: kiwi
Enter price: 70
```

```
Key and Values:
Key: grapes Value: 50
Key: pineapple Value: 80
Key: banana Value: 40
Key: apple Value: 60
Key: kiwi Value: 70

Total cost:
Total price:  300

Deleting associative array...
```

Exercise 6

1. Write a function that allows the user to select a directory from the list of directories. Move the selected directory to the first position of the list. (Using select statement).

```
kri@kri-ubuntu:~/workspace/uni8$ cat dir.sh
declare -a list
list=`ls f1`
echo $list
select dir in $list
do
        echo "Selected: $dir"
        echo $dir
        for i in $list
        do
                if [ $dir != $i ]
                then
                        echo $i
                fi
        done
done

kri@kri-ubuntu:~/workspace/uni8$ bash ./dir.sh
f11 f12 f13
1) f11
2) f12
3) f13
#? 1
Selected: f11
f11
f12
f13
#? 2
Selected: f12
f12
f11
```

```
f13
#? 3
Selected: f13
f13
f11
f12
#? ^C
```

2. Write a shell script to translate the contents of a file into Upper case, Lower case, title case and print not valid case when invalid argument passed where file name is entered through command line.(use select case).

```
kri@kri-ubuntu:~/workspace/uni8$ cat case.sh
if [ -e $1 ]
then
        select choice in UpperCase LowerCase TitleCase
        do
        case $choice in
                UpperCase) echo "File in Upper case: "
                        cat $1|tr "[a-z]" "[A-Z]"
                        ;;
                LowerCase) echo "File in Lower case: "
                        cat $1|tr "[A-Z]" "[a-z]"
                        ;;
                TitleCase) echo "Title case: "c
                        sed "s/\<./\U&/g" $1
                        ;;
                *) echo "Wrong choice"
        esac
        done
else
        echo "Invalid file"
fi

kri@kri-ubuntu:~/workspace/uni8$ bash ./case.sh newfile
1) UpperCase
2) LowerCase
3) TitleCase
#? 1
File in Upper case:
HI
BYE
THIS IS FUN
NO NOT FUN
TIME TO SLEEP
#? 2
File in Lower case:
hi
bye
```

```
this is fun
no not fun
time to sleep
#? 3
Title case: c
Hi
Bye
This Is Fun
No Not Fun
Time To Sleep
#? ^C
```