CET-211
Operating Systems
Experiment # 04

Experiment Title

Focusing on the usage of Test Commands and Conditional Statements

Assessment of CLO(s): 04

Performed on _____

Student Name:			
Roll No.		**Group**	
Semester		**Session**	

Total (Max)	**Performance (03)**	**Viva (03)**	**File (04)**	**Total (10)**
Marks Obtained				
Remarks (if any)				

Experiment evaluated by

Instructor's Name	**Engr. Bushra Aziz**		
Date		**Signature**	

Objective: To learn the usage of different types of operators and conditional statements.

THEORY

In bash and pdksh, a command called test is used to evaluate conditional expressions. You would typically use the test command to evaluate a condition that is used in a conditional statement or to evaluate the entrance or exit criteria for an iteration statement. The test command has the following syntax:

```
test
expression or
 [ expression ]
```

Several built-in operators can be used with the test command. These operators can be classified into four groups**: integer operators, string operators, file operators, and logical operators**.

The shell integer operators perform similar functions to the string operators except that they act on integer arguments. The following table lists the test command's integer operators.

The Test Command's Integer Operators

Operator	Meaning
Int1 -eq int2	Returns True if int1 is equal to int2.
Int1 -ge int2	Returns True if int1 is greater than or equal to int2.
Int1 -gt int2	Returns True if int1 is greater than int2.
Int1 -le int2	Returns True if int1 is less than or equal to int2.
Int1 -lt int2	Returns True if int1 is less than int2.
Int1 -ne int2	Returns True if int1 is not equal to int2.

The string operators are used to evaluate string expressions. The table below lists the string operators that are supported by the three shell programming languages.

The Test Command's String Operators

Operator	Meaning
Str1 = str2	Returns True if str1 is identical to str2.
Str1 != str2	Returns True if str1 is not identical to str2.
str	Returns True if str is not null.
-n str	Returns True if the length of str is greater than zero.
-z str	Returns True if the length of str is equal to zero.

The test command's file operators are used to perform functions such as checking to see if a file exists and checking to see what kind of file is passed as an argument to the test command. The following is the list of the test command's file operators.

The Test Command's File Operators

Operator	Meaning
-d filename	Returns True if file, filename is a directory.
-f filename	Returns True if file, filename is an ordinary file.
-r filename	Returns True if file, filename can be read by the process.
-s filename	Returns True if file, filename has a nonzero length.
-w filename	Returns True if file, filename can be written by the process.
-x filename	Returns True if file, filename is executable.

The test command's logical operators are used to combine two or more of the integer, string, or file operators or to negate a single integer, string, or file operator. The table below lists the test command's logical operators.

The Test Command's Logical Operators

Command	Meaning
! expr	Returns True if expr is not true.
expr1 -a expr2	Returns True if expr1 and expr2 are true.
expr1 -o expr2	Returns True if expr1 or expr2 is true.

The Test Command's Arithmetic Operators

Command	Meaning
$a + $b	Returns sum of two variables
$a - $b	Returns remaining value after subtraction
$a * $b	Returns product of two variables
$a / $b	Divide two variables
$a % $b	Return remainder after division of two variables

These test command arithmetic operators can be used in different contexts in shell scripts, often inside arithmetic expressions with expr, let, or $(( )) constructs.

Conditional Statements

The bash, pdksh, and tcsh each have two forms of conditional statements. These are the if statement and the case statement. These statements are used to execute different parts of your shell program depending on whether certain conditions are true. As with most statements, the syntax for these statements is slightly different between the different shells.

The if Statement

All three shells support nested if...then...else statements. These statements provide you with a way of performing complicated conditional tests in your shell programs. The syntax of the if statement is the same for bash and pdksh and is shown here: `if [ expression ] then commands elif [ expression2 ] then commands else commands fi`

The elif and else clauses are both optional parts of the if statement. Also note that bash and pdksh use the reverse of the statement name in most of their complex statements to signal the end of the statement. In this statement the fi keyword is used to signal the end of the if statement.

The elif statement is an abbreviation of else if. This statement is executed only if none of the expressions associated with the if statement or any elif statements before it were true. The commands associated with the else statement are executed only if none of the expressions associated with the if statement or any of the elif statements were true.

In tcsh, the if statement has two different forms. The first form provides the same function as the bash and pdksh if statement. This form of if statement has the following syntax:

```
if (expression1) then commands
else if (expression2) then
commands else commands
endif
```

Example:

This statement checks to see if there is a. profile file in the current directory:

```
if [ -f "sar.txt" ] then echo "There is a .txt file
in the current directory." else echo "Could not
find the .txt file."
fi
```

The case Statement

The case statement enables you to compare a pattern with several other patterns and execute a block of code if a match is found. The shell case statement is quite a bit more powerful than the case statement in Pascal or the switch statement in C. This is because in the shell case statement you can compare strings with wildcard characters in them, whereas with the Pascal and C equivalents you can compare only enumerated types or integer values.

Once again, the syntax for the case statement is identical for bash and pdksh and different for tcsh. The syntax for bash and pdksh is the following:

```
case string1 in
str1) commands;;
str2) commands;;
*) commands;
 ;
esac
```

String1 is compared to str1 and str2. If one of these strings matches string1, the commands up until the double semicolon (;;) are executed. If neither str1 nor str2 matches string1, the commands associated with the asterisk are executed. This is the default case condition because the asterisk matches all strings.

Example:

```
#!/bin/bash
echo "enter fruit name"
read n

case $n in
"apple")
echo "i am apple"
;;
"kiwi")
echo "green kiwi i am "
;;

"grapes")
echo "green and black grapes"
;;
*)
echo "invalid"
esac
```

Example:

The following code is an example of a bash or pdksh case statement. This code checks to see if the first command-line option was -i or -e. If it was -i, the program counts the number of lines in the file specified by the second command-line option that begins with the letter i. If the first option was -e, the program counts the number of lines in the file specified by the second command-line option that begins with the letter e. If the first command-line option was not -i or -e, the program prints a brief error message to the screen.

```
case $1 in
-i) count='grep ^i $2 | wc
     -l'  echo "The number of lines in $2 that start with an
     i is $count"
     ;;
-e) count='grep ^e $2 | wc -l' echo "The number of lines in
     $2 that start with an e is $count"
     ;;
* ) echo "That option is not
     recognized"
     ;; esac
```

Exercise:

1. What test command should be used to test that /usr/bin is a directory or a symbolic link?

2. Write a script that takes two strings as input compares them and depending upon the results of the comparison prints the results.
   The user may provide input to the Bash script using:
   read var

3. Write a script that takes a number (parameter) from 1-12 as input and uses case to display the name of corresponding month.

4. Write a script that takes command-line argument for age and marks , and decide whether student is eligible for admission or not.
   Eligibility Criteria:
   Age should be lesser than 18 and marks should be greater than 700