**Bahria University, Lahore Campus**

Department of Computer Sciences

Lab Journal 03

**(spring 2024)**

|  |  |  |
| --- | --- | --- |
| Course: | **Operating System Lab** | Date: 03-07-2024 |
| Course Code: | CSL-320 | Max Marks: 20 |
| Faculty’s Name: | Abdullah |  |

Name: Muhammad Hammad Enroll No: 03-134221-024

**Objective(s):**

Understanding of Shell Programming. Understanding of variables, loops, operators.

**Tool(s) used:**

Ubuntu, VIM Editor\Text Editor

**Lab Tasks :  
Task 1:** Write the output of programs for LINUX variables.

**Task 2:** Write a program to calculate the addition, subtraction, multiplication and division of numbers.

**Task 3.1:** Write a program that compares two numbers if a is greater than b it displays “a is greater than b”, otherwise it displays that ‘a is not equal to b’.

**Task 3.2:** Write a program that compares two numbers check whether the numbers are equal, a is greater than b or a is less than b.

**Task 4 :** Write a program using “case” that inputs a fruit from the user and displays “Apple pie” on the input of apple, “I like banana” on the input of banana and “New Zealand famous for kiwi” on the input of kiwi.

**Lab Grading Sheet :**

|  |  |  |  |
| --- | --- | --- | --- |
| **Task** | **Max Marks** | **Obtained Marks** | **Comments(*if any*)** |
| 1. | 05 |  |  |
| 2. | 05 |  |  |
| 3. | 05 |  |  |
| 4. | 05 |  |  |
| **Total** | **20** |  | **Signature** |

**Introduction**

Shell programming is a group of commands grouped together under single filename. After logging onto the system a prompt for input appears which is generated by a Command String Interpreter program called the shell. The shell interprets the input, takes appropriate action, and finally prompts for more input. The shell can be used either interactively - enter commands at the command prompt, or as an interpreter to execute a shell script. Shell scripts are dynamically interpreted, NOT compiled.

**Common Shells**

C-Shell - csh

The default on teaching systems Good for interactive systems Inferior programmable features.

Bourne Shell

bash or sh - also restricted shell – bsh (The Bourne Again Shell) It was written by Steve Bourne. Over the years the original Bourne Shell has been expanded, but it remains the basic shell provided in many commercial versions of Linux.

Korn Shell

It was written by David Korn This shell extended many features of Bourne Again Shell and added many new features.

Thomas C-Shell - tcsh

The TC Shell performs the same functions as Bourne Again Shell. It is an interactive command line interpreter as well as for high level programming languages.

Shell Keywords

echo, read, if fi, else, case, esac, for , while , do , done, until , set, unset, readonly, shift, export, break, continue, exit, return, trap , wait, eval ,exec, ulimit , umask.

**General Shell Terminologies**

The shebang line or hashbang#!

The “shbang” line is the very first line of the script and lets the kernel know what shall will be interpreting the lines in the script. The shbang line consists of #! Followed by the full pathname to the shell, and can be followed by options to control the behavior of shell.

Example

**#!/bin/bash**

Comments

Comments are descriptive material preceded by a # sign. They are ineffect until the end of a line and can be started anywhere on the line.

Example  
#This text is not interpreted by the shell.

**FIRST “HELLO WORLD” SHELL PROGRAM  
Step 1** Open VIM with the filename.sh extension.

**Step 2** Write the Hello World Program

#!bin/bash

#YOUR FIRST HELLO WORLD PROGRAM

echo ‘Hello World!’

**Step 3** Execution of SHELL Script:

By using change mode command

$ chmod u + x hello.sh

$ ./hello.sh

Shell variables

A variable is something to which we assign a value. The value assigned could be a number, text, or any data type.

1. **PATH** - Directory paths to search for commands.

1. **HOSTNAME** - The name of the computer.
2. **USER** - The user id of the user running this shell.
3. **SHELL** - The shell currently is used.
4. **TERM** - The type of terminal being used.
5. **PS1** - The prompt to print when then shell is ready for another command.

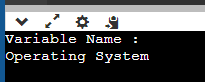
**Task 1:**

Write the output of the below programs.

**i)** #!/bin/bash  
 #Variable Assignment & Accessing   
 echo "Variable Name : "  
 Name="Operating System"

echo $Name

**Output:**



ECHO Statement

Similar to the output statement. To print output to the screen, the echo command is used.

**Syntax:** Echo “String” (or) echo $ b (for variable).

**Example:** echo "What is your name?"

READ Statement

To get the input from the user.

**Syntax:** read x y (no need of commas between variables)

**Reading user input:** The read command takes a line of input from the user and assigns it to a variable(s) on the right-hand side. The read command can accept multiple variable names. Each variable will be assigned a word.

**ii)** #!/bin/bash   
 #Input from user   
 echo "Enter your name"

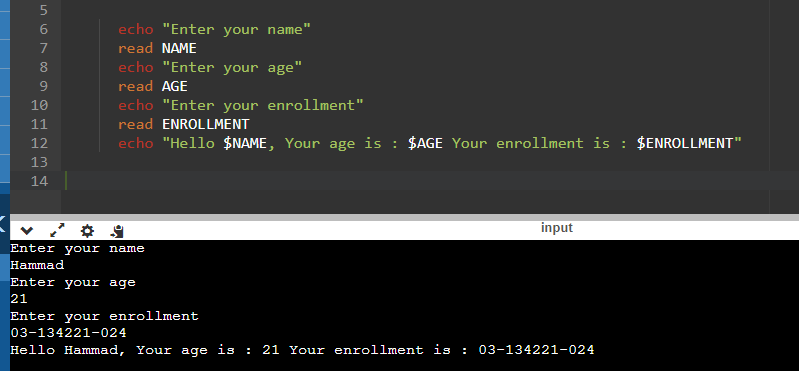
read NAME

echo "Enter your age"

read AGE  
 echo "Enter your enrollment"  
 read ENROLLMENT  
 echo "Hello $NAME, Your age is : $AGE Your enrollment is : $ENROLLMENT**"**

page6image43302912

**OUTPUT**



**iii)** #!/bin/bash #readonly Variables

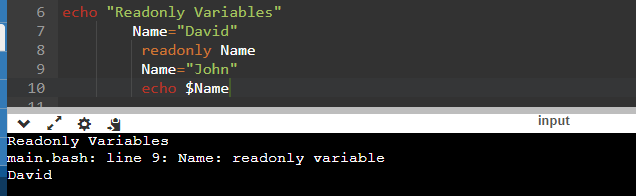
echo "Readonly Variables"

Name=”David”

readonly Name

Name=’John’

**OUTPUT**



**iv)** #!/bin/bash

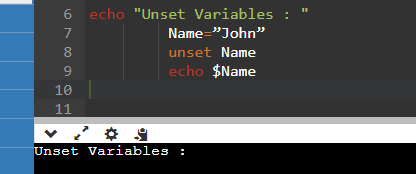
echo "Unset Variables : "

Name=”John”

unset Name

echo $Name

**OUTPUT**



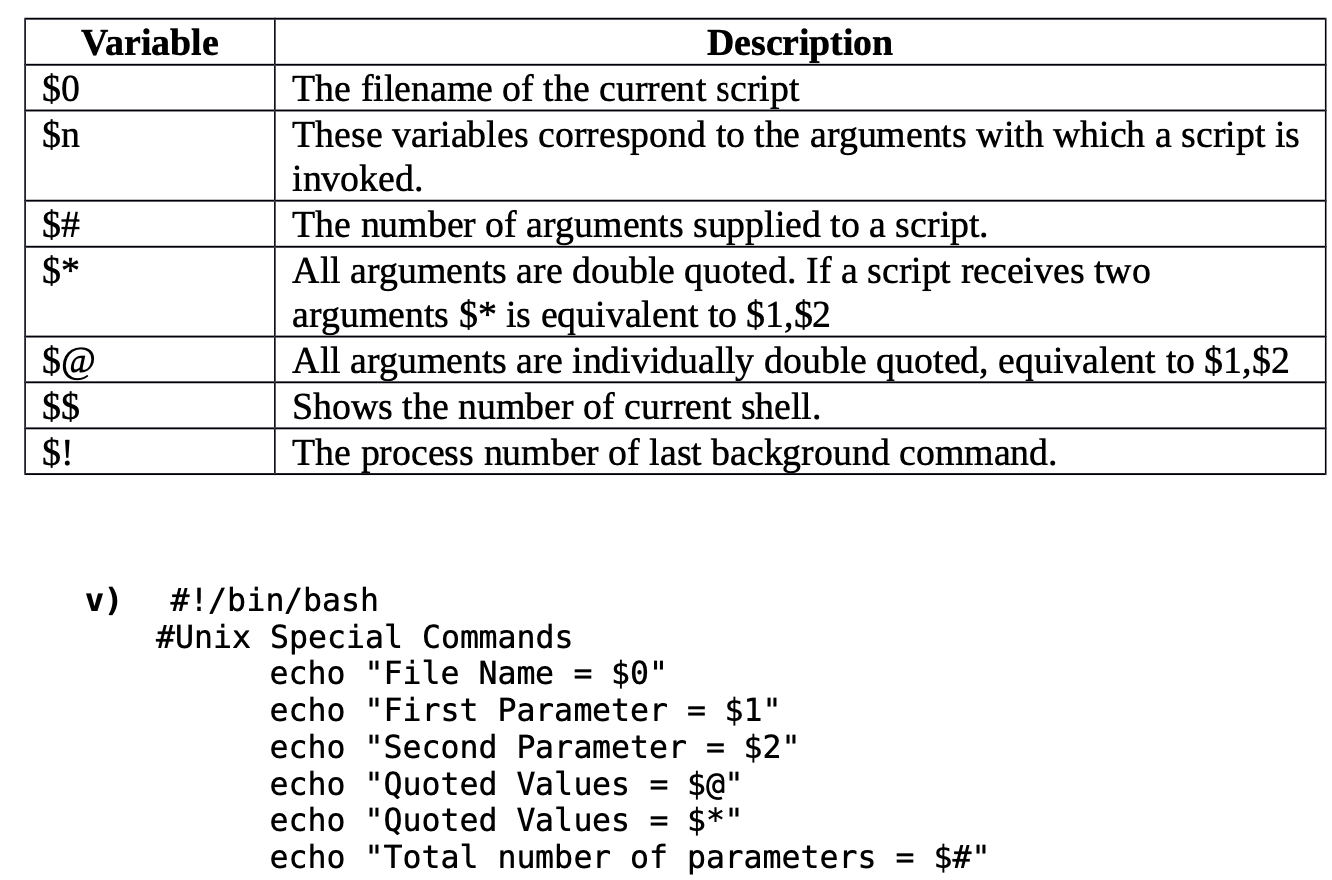
**Wildcards**

There are some characters that are evaluated by the shell in a special way. They are called shell meta characters or “Wildcards. These characters are neither number nor letters.

Example

\*,?,[ ],$  
$$echo $$ -- It represents process ID Number, or PID of the current shell

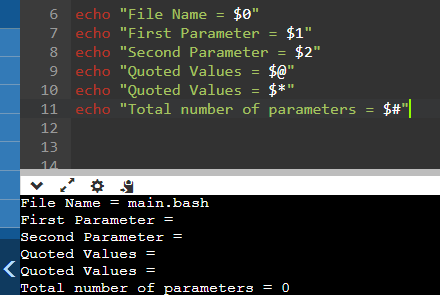
The following table shows a number of special variables that can be used in shell scripts.



**OUTPUT**

**vi)** #!/bin/bash #Special Commands   
echo "File Name = $0"  
echo "First Parameter = $1"  
echo "Second Parameter = $2"  
echo "Quoted Values = $@"  
echo "Quoted Values = $\*"  
echo "Total number of parameters = $#"  
echo $?

**OUTPUT**



**EXPRESSION Command**

Arithmetic Operations

To perform all arithmetic operations. The Bourne shell does not support arithmetic. LINUX/Linux commands must be used to perform calculations.

**Syntax:**



**Task 2** Write a program for calculating addition, subtraction, multiplication and division of two   
numbers.

**Program:**

echo "Enter First Number : "

read var1

echo "Enter Second Number : "

read var2

echo "Enter 1 For addition,Enter 2 For Subtraction,Enter 3 For Division,Enter 4 For Multiplication"

read x

if [[ x -eq 1 ]]; then

echo "Sum of $var1 and $var2 is $((var1 + var2))"

elif [[ x -eq 2 ]]; then

echo "Sub of $var1 and $var2 is $((var1 - var2))"

elif [[ x -eq 3 ]]; then

echo "Division of $var1 and $var2 is $((var1 / var2))"

elif [[ x -eq 4 ]]; then

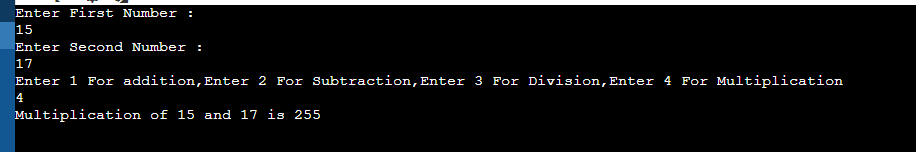
echo "Multiplication of $var1 and $var2 is $((var1 \* var2))"

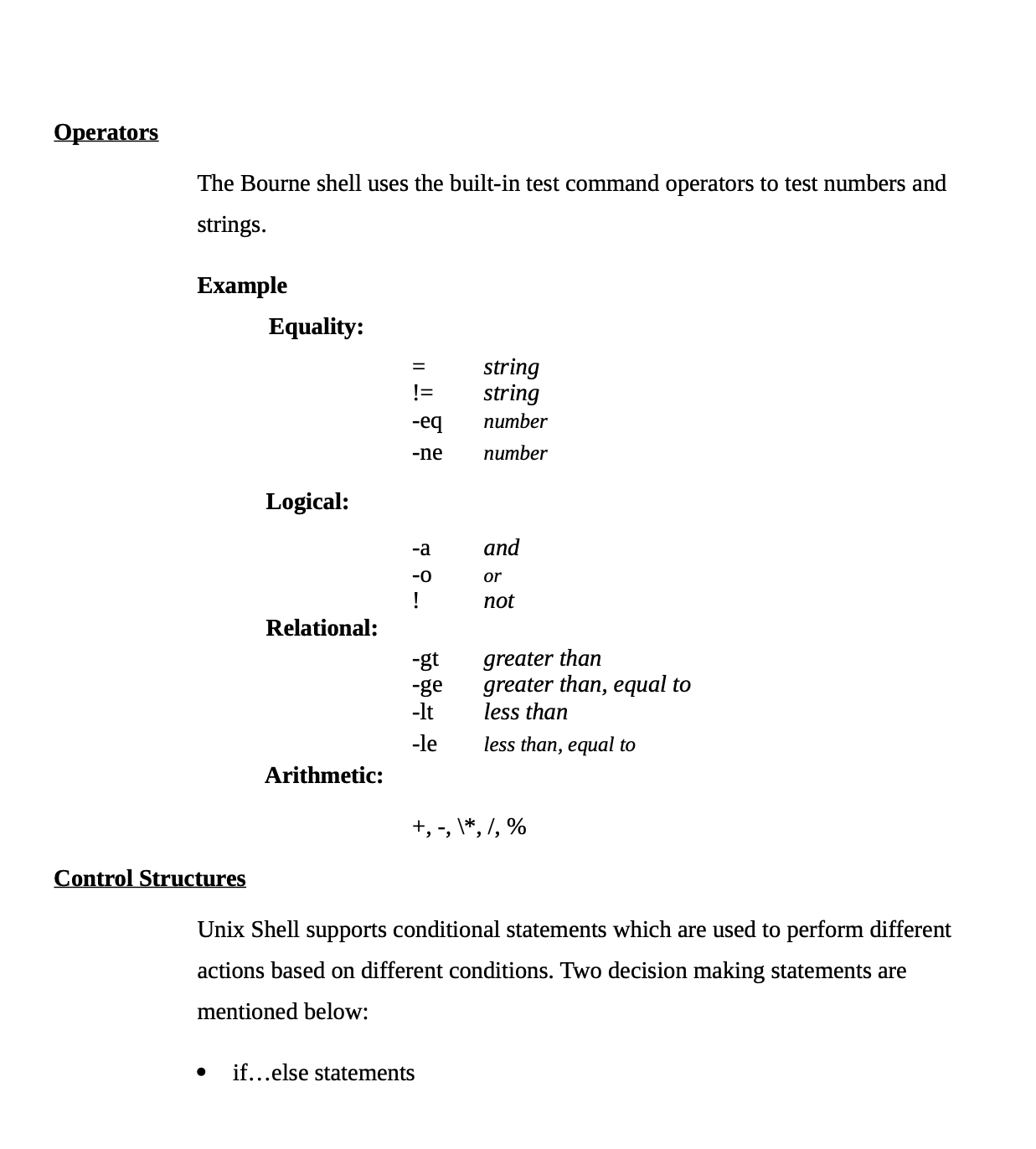
else

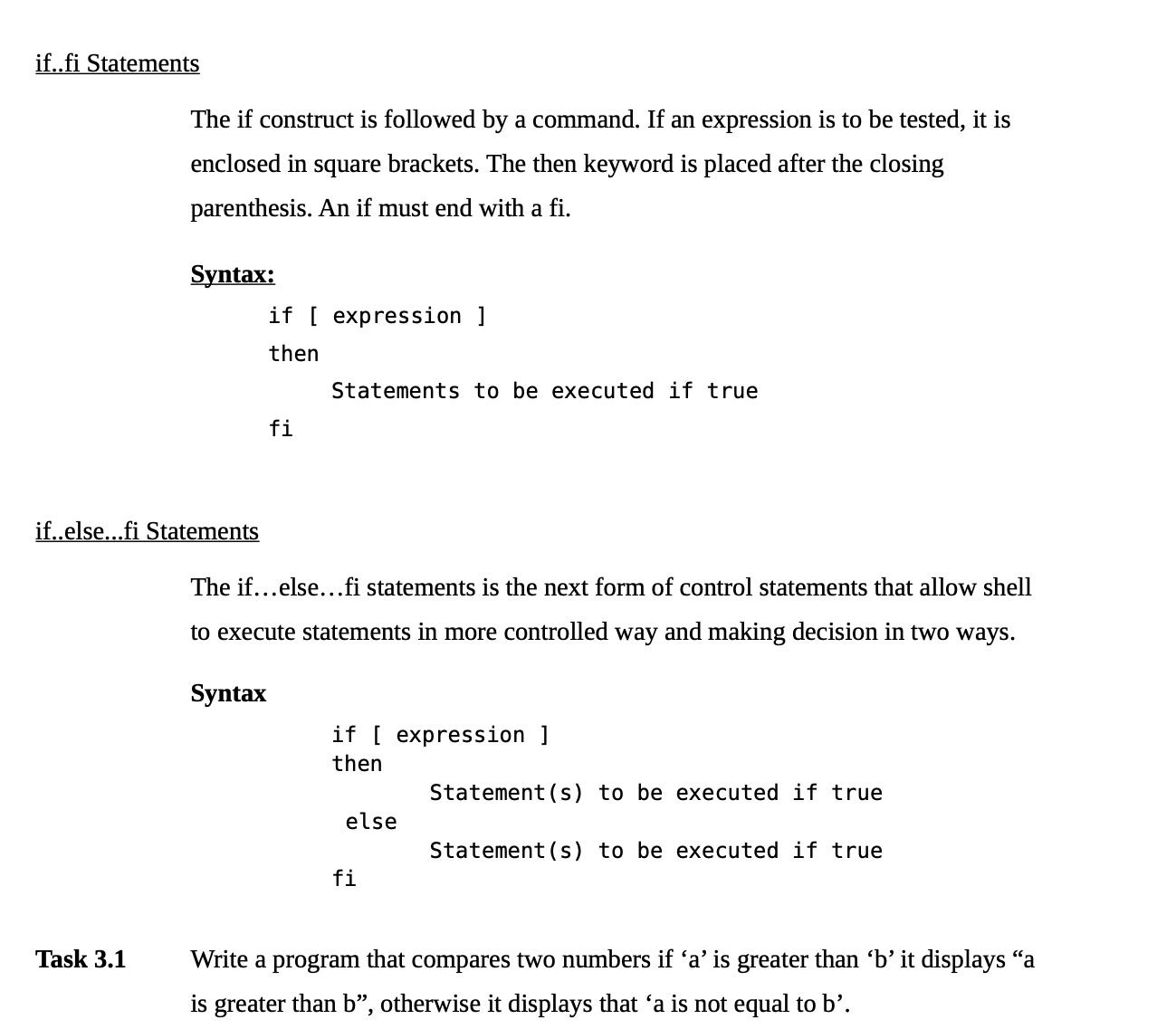
echo "Multiplication of $var1 and $var2 is $((var1 \* var2))"

fi

**Output:**



****

****

**Program:**

echo "Enter the value of a:"

read a

echo "Enter the value of b:"

read b

if [ "$a" -gt "$b" ]; then

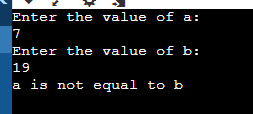
echo "a is greater"

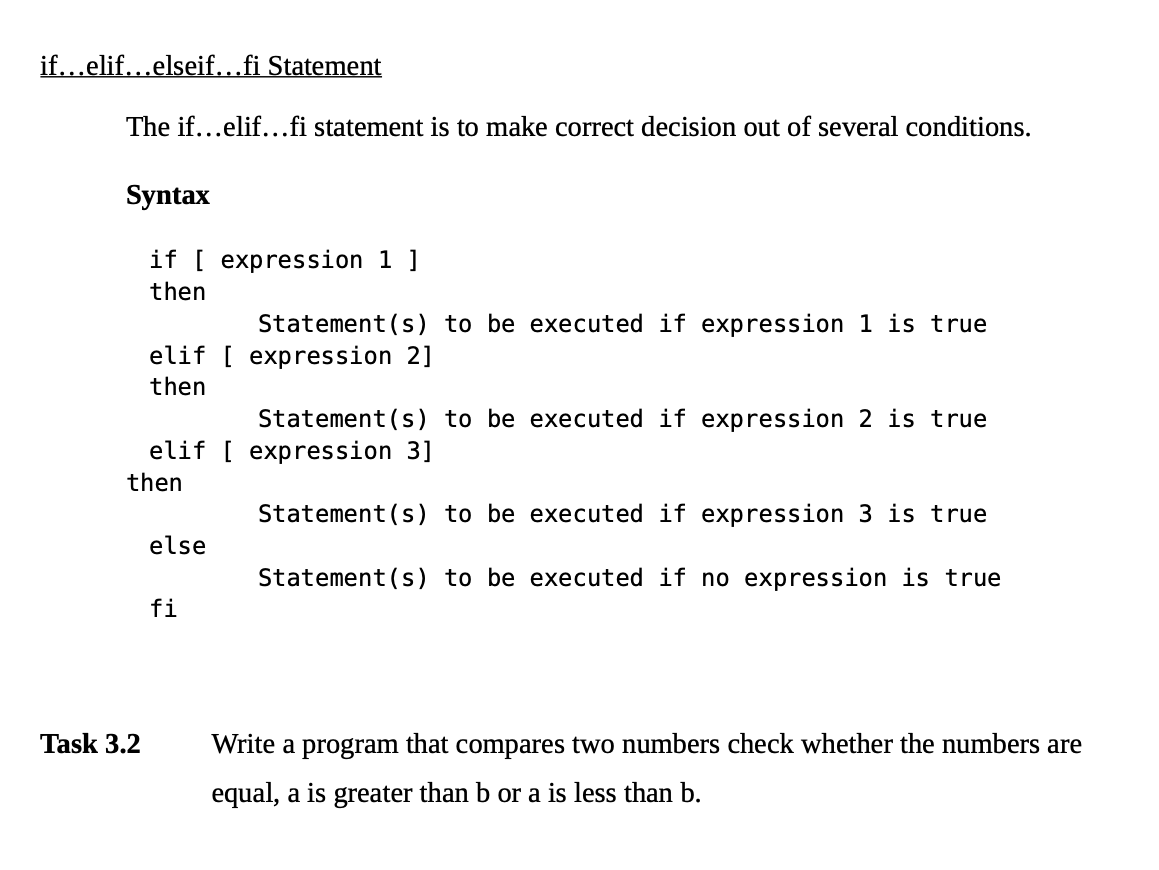
else

echo "a is not equal to b"

fi

**Output:**



****

**Program:**

echo "Enter the value of a:"

read a

echo "Enter the value of b:"

read b

if [ $a -gt $b ]; then

echo "a is greater than b"

elif [ $a -eq $b ]; then

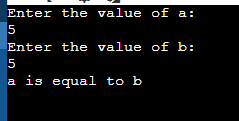
echo "a is equal to b"

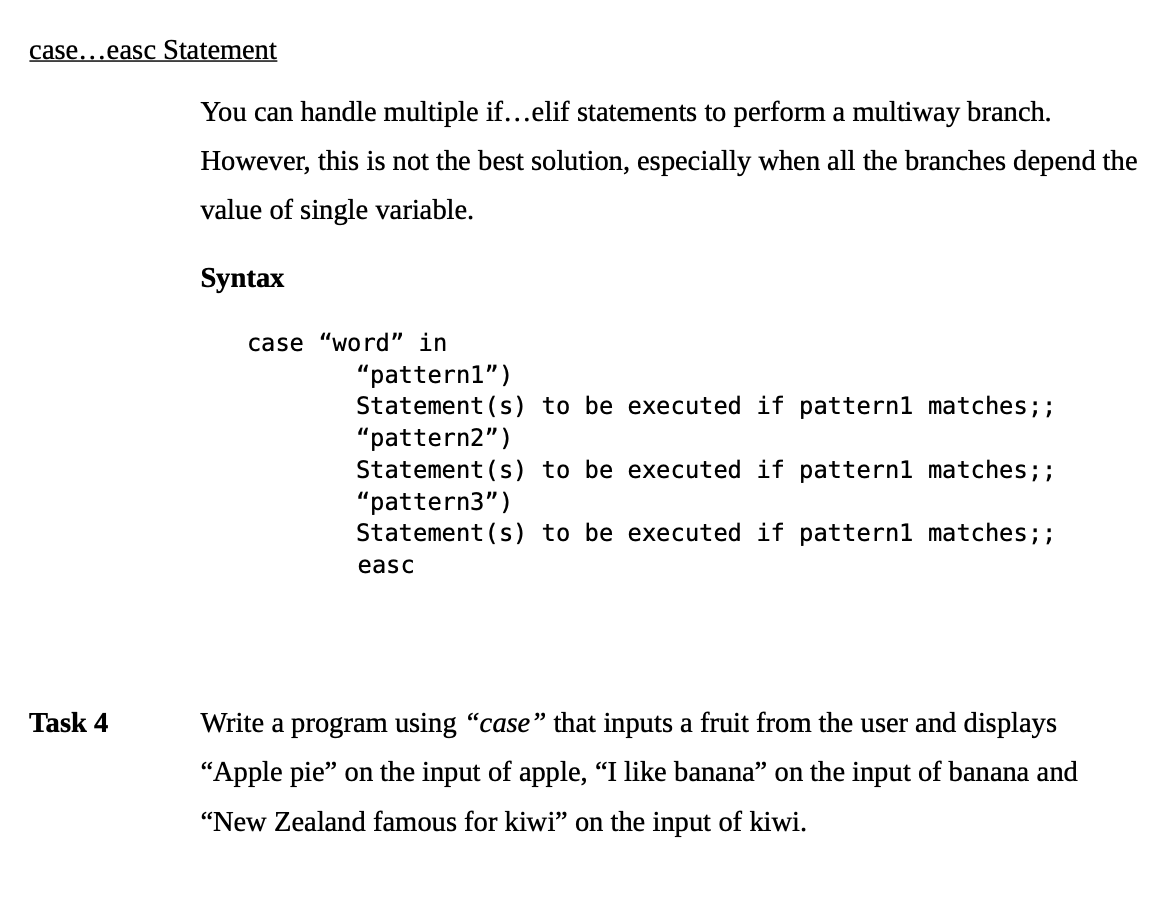
else

echo "a is lesser than b"

fi

**Output:**



****

**Program:**

echo "Enter the case:"

read a

if [ $a = apple ]; then

echo "Apple Pie"

elif [ $a = banana ]; then

echo "I like banana"

elif [ $a = kiwi ]; then

echo "New Zeland famous for kiwi"

else

echo "Wrong Input"

fi

**Output:**

