ASSIGNMENTS for Batch R2

- **What will be the output of the following programs?**

Example 1:

$ echo "enter a number"
enter a number
$ read a
32
$ echo "enter another number"
$ read b
9
$ var = $((a + b))
$ echo $var
41 / Output */*
$

Example 2:

$ i=1
$ while [[$i -le 10]]
> do
> echo $i
> ((i += 1))
> done
1 / Output */*
2
3
4
5
6
7
8
9
10
$

Example 3:

$ i=1
$ until [[$i -gt 10]]
> do
> echo $i
> ((i += 1))
> done

1 / Output */*
2
3
4
5
6
7
8
9
10
$

Example 4:
$ for i in {1..5}
> do
> echo $i
> done
1 / Output */*
2
3
4
5
$

Example 5:
$ sum=0
$ for i in 1 2 3 4 5
> do
> ((sum += i))
> echo $sum
> done
1 / Output */*
3
6
10
15
$

Example 6:

```
$ sum=0
$ for i in 1 2 3 4 5
> do
> if [ $i -eq 4]
> then
> break
>fi              /* end of if */
> (( sum += i))
> echo $sum
> done      /* end of for */
1         /* Output */
3
6
$
```

Example 7:

```
$ sum=0
$ for i in 1 2 3 4 5
> do
> if [ $i -eq 4]
> then
> continue
>fi              /* end of if */
> (( sum += i))
> echo $sum
> done      /* end of for */
1         /* Output */
3
6
11
$
```

- *Syntax of the **case...esac** statement*

```
case <word> in

<first pattern>)
        <statements>
        ;;
<second pattern>)
        <statements>
        ;;
    *)
        <default statements>
        ;;
esac                                    /* end of case statement */
```

Example 8:

```
$ echo "enter a number"
enter a number
$ read num
4
$ case $num in
> 1) echo you entered 1
> ;;
> 2) echo you entered 2
> ;;
> 4) echo you entered 4
> ;;
> 5) echo you entered 5
> ;;
> *) echo your entry does not match  /* default statement */
> ;;
> esac      /* end of case */
you entered 4      /* Output */
$
```

Exercise 1: *Write a program to print all prime numbers from 1 to 200.*
(Hint: Use nested loops, ***break*** *and* ***continue****)*

Exercise 2: *Write a program to print the first 'n' terms in the Tribonacci series, where the value for 'n' is given as an input through the keyboard.*
(Tribonacci series : 0 1 1 2 4 7 13 24 44 ……..)

Exercise 3: *Write a program to check whether a given number is Krishnamurthy number or not.*

Note: A Krishnamurthy number is a number whose sum of the factorial of digits is equal to the number itself. For example 145, sum of factorial of each digits:
1! + 4! + 5! = 1 + 24 + 120 = 145

Exercise 4: Write a program that asks the user to enter a character and then determines whether the user entered a smallcase letter, a capital letter, a digit or a special symbol.

- ***Exercises on Strings:***

Note: While comparing two strings there is a space on either side of '='. This is necessary, otherwise it would become a simple assignment.

Exercise 5: Declare two string variables ***str1*** and ***str2***, assign a string "VNIT" to the variable ***str1*** and a string "Nagpur" to the variable ***str2***. Print the two strings and their length using the variables under **$ prompt**. Concatenate the strings (using ***str1*** and ***str2***) to the third variable ***str3***, print the concatenated string and its length by ***str3***.

Exercise 6: Write a shell script to ***reverse*** a string.

Exercise 7: Write a shell script that will enter a line of text containing a word, a phrase or a sentence, and determine whether or not the text is a ***palindrome***.

- ***Functions:***

To declare a function, simply use the following Syntax:

```
function_name()
{
     List of commands
}
```

Note: We can have DOS like commands for their Unix equivalents.

Example 1:

```
$ dir()
> {
> ls
> }
$ dir            /* Invoke the function */
                  /* Lists all files and directories using dir */
```

- ***Pass parameters to a function***

You can define a function that will accept parameters while calling the function. These parameters would be represented by $1, $2 and so on.

Example 2:

```
$ fn1()
> {
> echo “Hello $1 $2”
> }
$ fn1 Good Morning          /* Invoke the function */
Hello Good Morning          /* Output */
$
```

- ***Returning a value from a function***

If you execute an ***exit*** command from inside a function, its effect is not only to terminate execution of the function but also of the shell program that called the function.

Based on the situation you can return any value from your function using the ***return*** command whose syntax is as follows:

return *value*

Example 3:

```
$ fn2()
> {
> echo “Hello $1 $2”
> return 100
> }
$ fn2 Good Morning          /* Invoke the function */
Hello Good Morning          /* Output */
$ ret=$?                    /* Capture the value returned by the last command */
$ echo “The return value is $ret”
The return value is 100     /* Output  */
$
```

- ***Nested Functions***

One of the more interesting features of functions is that they can call themselves and also other functions. A function that calls itself is known as a recursive function.

/* Calling one function from another */

Example 4:

```
$ fn3()
> {
> echo “This is the first function”
> fn4
> }
$ fn4()
> {
> echo “This is the second function”
> }
$ fn3                       /* Invoke the function */
This is the first function  /* Output */
This is the second function /* Output */
$
```

Exercise 8: Define a function that computes the sum of first *n* odd integers. Write a shell script to call the function with a value for the parameter *n*, and display the result.

Note: *n* = 3, sum = 1 + 3 + 5 = 9

- ***Recursive function***

/* compute the sum of first '*n*' odd integers *******/*
Example 5:
$ computeOddSum()
> {
> if (($1 <= 1))
> then
> sum=1
> echo 1
> else
> sum=$(computeOddSum $(($1 - 1)))
> lastTerm=$((2 * $1 - 1))
> echo $(($sum + $lastTerm))
> fi
> }
$ computeOddSum 5 /* Invoke the function to compute the sum of first 5 odd integers*/
/* Here function parameter $1 gets value 5*/
25 /* Output i.e., 1 + 3 + 5 + 7 + 9 = 25 */
$

Exercise 9: Write a shell script to compute the factorial value of an integer entered through the keyboard (Use a **recursive function**).

- **Arrays in Shell Scripting:**

An array is a systematic arrangement of the same type of data. But in shell script, the values contained in an Array may be of same type or different types, since by default in shell script, everything is treated as a string. An array indexing starts from 0.

We can declare an array in a shell script in different ways. Here is an example.

Example 1:
$ arr=(10 30 50 22 55 77 11) /* Array declaration with some values */
$ echo ${arr[*]} /* To print all elements of the array */
10 30 50 22 55 77 11 /* Output */
$ echo ${arr[@]} /* To print all elements of the array */
10 30 50 22 55 77 11 /* Output */

$ echo ${arr[@]:3} /* To print all elements from a particular index */
22 55 77 11 /* Output */

$ echo ${arr[@]:0} /* To print all elements from index 0 */
10 30 50 22 55 77 11 /* Output */

$echo ${arr[4]} /* To print fifth element */
55 /* Output */

Example 2:
$ read a[0]
32 /* Input */
$ read a[1]
44 /* Input */
$ read a[2]
25 /* Input */
$ read a[3]
66 /* Input */

$ echo ${a[@]} /* To print all elements of the array */
32 44 25 66 /* Output */

Example 3:
$ a=(3 34 24 64 14 44 49) /* Array declaration with some values */
$ echo ${a[@]} /* To print all elements of the array */
3 34 24 64 14 44 49 /* Output */
$ fnarray() /* Define a function to print all elements of the array **a** from an index represented by $1 */
> {
> echo ${a[@]:$1}
> }
$ fnarray 4 /* Invoke the function to print array elements from index 4, the parameter $1 represents the value 4 */
14 44 49 /* Output */

Exercise 10: Consider an array of *n* integers where *n* is the size of the array. Write a shell script to find the maximum value among *n* integers stored in the array (Use a **recursive function**).

Exercise 11: Consider an array of *n* integers where *n* is the size of the array. Write a shell script to find the maximum value among *n* integers stored in the array **(Without using a recursive function)**.

Exercise 12: Write a shell script to generate Fibonacci numbers less than or equal to '*n*' *where the value for 'n' is given as an input through the keyboard* (Use a **recursive function**).
(Fibonacci series : 0 1 1 2 3 5 8 13 21 34 ……..)

Note: If n = 35 then output should be 0 1 1 2 3 5 8 13 21 34