CS315
HOMEWORK 3 REPORT

FATİH SEVBAN UYANIK
21602486

Question 1

By defining optional parameters, at the beginning, the programmer is able to add mandatory parameters. After, for adding optional parameters, **#!optional** needs to be declared. After this declaration, optional parameters can be added as many as the programmer wants. Below are some sample functions and function calls.

Example Function 1

In this function, only optional parameters are available and the function tries to print them. If a value for the parameters is not assigned while calling the function, the parameters that does not get a value will return or print #f.

Example Function Definition and Syntax

```
(define (example-optional-1 #!optional param1 param2 param3)
    (print "------------------\n")
    (print "Param1: " param1 "\n")
    (print "Param2: " param2 "\n")
    (print "Param3: " param3 "\n")
    (print "------------------\n"))
```

Function Calls

no parameter values are passed.
(example-optional-1)

only for the first parameter is a value passed.
(example-optional-1 100)

For the first and second
parameter is value passed.
(example-optional-1 100 200)

for all of the parameters, values are passed.
(example-optional-1 100 200 300)

Result of Call

```
----------EXAMPLE FUNCTION1----------
------------------
Param1: #f
Param2: #f
Param3: #f
------------------
------------------
Param1: 100
Param2: #f
Param3: #f
------------------
------------------
Param1: 100
Param2: 200
Param3: #f
------------------
------------------
Param1: 100
Param2: 200
Param3: 300
------------------
------------------------------------------
```

Example Function 2

In this function, there is one actual parameters which is mandatory and 3 optional parameters. The function tries to print them. The mandatory parameter needs to get a value while function call. If a value for the optional parameters is not assigned while calling the function, the parameters that does not get a value will print #f.

Example Function Definition and Syntax

```
(define (example-optional-2 param1 #!optional param2 param3 param4)
    (print "------------------\n")
    (print "Param1: " param1 "\n")
    (print "Param2: " param2 "\n")
    (print "Param3: " param3 "\n")
    (print "Param4: " param4 "\n")
    (print "------------------\n"))
```

Function Calls

For the actual parameter,
a value is passed.
(example-optional-2 100)

For the actual parameter, a value
is passed and for the first optional
parameter, a value is passed.
(example-optional-2 100 200)

For the actual parameter, a value
is passed and for the first two
optional parameters, values are passed.
(example-optional-2 100 200 300)

For all of the parameters, values are passed.
(example-optional-2 100 200 300 400)

Result of Call

```
----------EXAMPLE FUNCTION2----------
------------------
Param1: 100
Param2: #f
Param3: #f
Param4: #f
------------------
------------------
Param1: 100
Param2: 200
Param3: #f
Param4: #f
------------------
------------------
Param1: 100
Param2: 200
Param3: 300
Param4: #f
------------------
------------------
Param1: 100
Param2: 200
Param3: 300
Param4: 400
------------------
------------------------------------------
```

Question 2

After defining optional parameters, we can also assign default values to them. Default values are assigned by opening additional parentheses which indicate the specific value. The example functions below illustrate how to use default values.

Example Function 1

In this function, only optional parameters are available and the function tries to print them. In addition, for each optional parameter, default value 0 is assigned. If a value for the optional parameters is not assigned while calling the function, the parameters that does not get a value will return or print their assigned default values which is 0.

Example Function Definition and Syntax

```
(define (example-default-1 #!optional (param1 0) (param2 0) (param3 0))
    (print "------------------\n")
    (print "Param1: " param1 "\n")
    (print "Param2: " param2 "\n")
    (print "Param3: " param3 "\n")
    (print "------------------\n"))
```

Function Calls

In this function call, no value is assigned to the optional parameters.
(example-default-1)

Only for the first optional parameter, a value is passed.
(example-default-1 100)

Only for the first two optional parameters, values are passed.
(example-default-1 100 200)

Only for the first three optional parameters, values are passed.
(example-default-1 100 200 300)

Result of Call

```
----------EXAMPLE FUNCTION1----------
----------
Param1: 0
Param2: 0
Param3: 0
----------
----------
Param1: 100
Param2: 0
Param3: 0
----------
----------
Param1: 100
Param2: 200
Param3: 0
----------
----------
Param1: 100
Param2: 200
Param3: 300
----------
----------
```

Example Function 2

In this function, only optional parameters are available and the function tries to print them. In addition, except the first parameter, default value 0 is assigned to optional parameters. If no value is passed for a parameter, then it will print its default value. If the parameter does not have a default value, it will print the value of #f.

Example Function Definition and Syntax

```
(define (example-default-2 #!optional param1 (param2 0) (param3 0))
    (print "------------------\n")
    (print "Param1: " param1  "\n")
    (print "Param2: " param2 "\n")
    (print "Param3: " param3 "\n")
    (print "------------------\n"))
```

Function Calls

In this function call, no value is assigned to the optional parameters.
(example-default-2)

Only for the first optional parameter, a value is passed.
(example-default-2 100)

Only for the first two optional parameters, values are passed.
(example-default-2 100 200)

Only for the first three optional parameters, values are passed.
(example-default-2 100 200 300)

Result of Call

```
----------EXAMPLE FUNCTION2----------
----------
Param1: #f
Param2: 0
Param3: 0
----------
----------
Param1: 100
Param2: 0
Param3: 0
----------
----------
Param1: 100
Param2: 200
Param3: 0
----------
----------
Param1: 100
Param2: 200
Param3: 300
----------
----------
```

Question 3

The parameter "param1" is an actual parameter and gets the first argument, whereas, the parameter "param2" gets the rest of the passed values as a list.

Example Function 1

Prints the actual parameter(param1) as it is and the rest parameter(param2) as a list.

Example Function Definition and Syntax

```
(define (example-rest-1 param1 . param2)
    (print "------------------\n")
    (print "Param1: " param1 "\n")
    (print "Param2: ")
    (printList param2)
    (print "------------------\n"))
```

Function Calls

In this function call, the first parameter 10
is counted as param1 and the rest parameter
param2 gets the remaining parameters as list.
(example-rest-1 10 70 20)

In this function call, the first parameter 80
is counted as param1 and the rest parameter
param2 gets the remaining parameter as list.
(example-rest-1 80 70)

In this function call, the first parameter 90
is counted as param1 and the rest parameter param2
will be an empty list because no values are passed.
(example-rest-1 90)

In this function call, the first parameter 10
is counted as param1 and the rest parameter
param2 gets the remaining parameters as list.
(example-rest-1 90 50 66 45 22 35 69)

Result of Call

----------EXAMPLE FUNCTION1----------

Param1: 10
Param2: 70 20

Param1: 80
Param2: 70

Param1: 90
Param2:

Param1: 90
Param2: 50 66 45 22 35 69

Question 4

In lisp keyword parameters can also be added. Before declaring keyword parameters, #!key needs to be added. Parameters that contain a key needs to be called with their corresponding key. The followings are examples of keyword parameters.

Example Function 1

In this function, param1 and param2 are actual parameters and param3 and param4 are parameters containing key. These parameters needs to be called with their corresponding key.

Example Function Definition and Syntax

```
(define (example-key-1 param1 param2 #!key param3 param4)
    (print "--------------------\n")
    (print "Param1 : " param1  "\n")
    (print "Param2 : " param2 "\n")
    (print "Param3 : " param3 "\n")
    (print "Param4 : " param4 "\n")
    (print "--------------------\n"))
```

Function Calls

This function passes values for only actual parameters because it does not uses any keys for the parameters containing a key. The parameters containing key will output #f.

(example-key-1 100 200)

This function passes values for actual and key parameters.

(example-key-1 100 200 param3: 300)

This function passes values for actual and key parameters.

(example-key-1 100 200 param3: 300 param4: 400)

Result of Call

----------EXAMPLE FUCTION1----------

Param1 : 100
Param2 : 200
Param3 : #f
Param4 : #f

Param1 : 100
Param2 : 200
Param3 : 300
Param4 : #f

Param1 : 100
Param2 : 200
Param3 : 300
Param4 : 400

Example Function 2

In this function, param1 and param2 are actual parameters and param3 and param4 are parameters containing key. In addition, a default value is assigned for the key parameter param3.

Example Function Definition and Syntax

```
(define (example-key-2 param1 param2 #!key (param3 1000) param4)
    (print "-------------------\n")
    (print "Param1 : " param1  "\n")
    (print "Param2 : " param2 "\n")
    (print "Param3 : " param3 "\n")
    (print "Param4 : " param4 "\n")
    (print "-------------------\n"))
```

Function Calls

This function passes values for only actual parameters because it does not uses any keys for the parameters containing a key. The parameters containing key will output #f.

(example-key-2 100 200)

This function passes values for actual and key parameters.

(example-key-2 100 200 param4: 300)

This function passes values for actual and key parameters.

(example-key-2 100 200 param3: 300 param4: 400)

Result of Call

----------EXAMPLE FUCTION2----------

Param1 : 100
Param2 : 200
Param3 : 1000
Param4 : #f

Param1 : 100
Param2 : 200
Param3 : 1000
Param4 : 300

Param1 : 100
Param2 : 200
Param3 : 300
Param4 : 400

