CS315

HOMEWORK 2 REPORT

FATİH SEVBAN UYANIK 21602486

JavaScript

a) What are the location of tests in Logically Controlled Loops ? Pretest, posttest, or both?

In JavaScript, there are two Logically Controlled Loops which are **while** and **do-while** loops.

While Loops

```
num = 10
pretestFlag = true

while (num < 10) {
    pretestFlag = false
    results += "<pre>While Loop in JavaScript is Post Test<br></pre>"
    results += "<pre>Value in the loop is " + num + "<br></pre>"
    num += 1
}

if (pretestFlag)
    results += "<pre>While Loop in JavaScript is Pre Test<br></pre>"

results += "<pre>-----------------------------------------------------------------<br></pre>"
results += "<br><br>"
```

This code snippet prints out:

While Loop in JavaScript is Pre Test

If JavaScript While Loop would have post test, then no matter what num is, it would execute the while loop at least once, hence, pretTestFlag variable would be false and the program would output Post Test condition which is stated inside the loop. Since JavaScript while loop has pre test, the check is done before the execution of the loop and because num is 10, the loop is not executed and it is stated that Javascript while loop is pretest by the program.

As a conclusion, **JavaScript while loop has pre test.**

Do - while Loops

```
num = 10
pretestFlag = true

do {
    pretestFlag = false
    results += "<pre>Do-While Loop in JavaScript is Post Test<br></pre>"
    results += "<pre>Value in the loop is " + num + "<br></pre>"
    num += 1
} while (num < 10)

if (pretestFlag)
    results += "<pre>Do-While Loop in JavaScript is Pre Test<br></pre>"

results += "<pre>-------------------------------------------------------------<br></pre>"
results += "<br><br>"
```

This code snippet prints out:

Do-While Loop in JavaScript is Post Test
Value in the loop is 10

If JavaScript Do-While Loop would have pre test, then the test or check would be done before the execution of the loop and since num is 10, the do-while loop would not execute. However, because JavaScript Do-While loop is post test, no matter what num is, it executes the while loop at least once. That's why, it prints out that Javascript Do-While loop is post test and the value in the loop is 10.

As a conclusion, **JavaScript Do-While loop has post test.**

b) What kind of user-located loop control mechanisms are provided in JavaScript?

In JavaScript, there are two user-located loop control mechanisms which are **break** & **continue**.

Break

```
index = 10
while (index > 0) {
    results += "<pre>" + index + "<br></pre>"
    index -= 1

    // Breaks or exits the loop.
    if (index == 8)
        break
}
```

This code snippet prints out:

10
9

When the index variable is equal to 8, break statement is executed and it results in breaking or exiting the loop. That way, programmers are able to exit a loop in whichever specific condition they want.

Continue

```
index = 10
while (index > 0) {
    // Passes to next iteration immedately
    // for the values of 8 and 5.
    if (index == 8 || index == 5) {
        index -= 1
        continue
    }

    results += "<pre>" + index + "<br></pre>"
    index -= 1
}
```

This code snippet prints out:

10
9
7
6
4
3
2
1

When the index variable is equal to 8 and 5, continue statement is executed and this statement passes to the next iteration or cycle immediately. After executing the continue statement, the rest of the code in the loop is not executed for the specified cycle and next cycle starts to execute immediately. That way, programmers are able to manipulate a loops functionality by passing to next cycles.

PHP

a) What are the location of tests in Logically Controlled Loops ? Pretest, posttest, or both?

In PHP, there are two Logically Controlled Loops which are **while** and **do-while** loops.

While Loops

```
$num = 10;
$preTestFlag = true;

while( $num < 10 ) {
    $preTestFlag = false;
    print("<pre>While Loop in PHP is Post Test<br></pre>");
    $num++;
}

if ($preTestFlag)
    print("<pre>While Loop in PHP is Pre Test<br></pre>");

print("<pre>-----------------------------------------------------------------<br></pre>");
print("<br><br>");
```

This code snippet prints out:

While Loop in PHP is Pre Test

If PHP While Loop would have post test, then no matter what num is, it would execute the while loop at least once, hence, pretTestFlag variable would be false and the program would output Post Test condition which is stated inside the loop. Since PHP while loop has pre test, the check is done before the execution of the loop and because num is 10, the loop is not executed and it is stated that PHP while loop is pretest by the program.

As a conclusion, **PHP while loop has pre test.**

Do - while Loops

```php
$num = 10;
$preTestFlag = true;

do {
    $preTestFlag = false;
    print("<pre>Do-While Loop in PHP is Post Test<br></pre>");
    print("<pre>Value in the loop is " . $num . "<br></pre>");
    $num++;
} while ($num < 10);

if ($preTestFlag)
    print("<pre>Do-While Loop in PHP is Pre Test<br></pre>");

print("<pre>-----------------------------------------------------------------<br></pre>");
print("<br><br>");
```

This code snippet prints out:

Do-While Loop in PHP is Post Test
Value in the loop is 10

If PHP Do-While Loop would have pre test, then the test or check would be done before the execution of the loop and since num is 10, the do-while loop would not execute. However, because PHP Do-While loop is post test, no matter what num is, it executes the while loop at least once. That's why, it prints out that PHP Do-While loop is post test and the value in the loop is 10.

As a conclusion, **PHP Do-While loop has post test.**

b) What kind of user-located loop control mechanisms are provided in PHP?

In PHP, there are two user-located loop control mechanisms which are **break** & **continue**.

Break

```php
$index = 10;
while ($index > 0) {
    print("<pre>" . $index . "<br></pre>");
    $index--;

    # Breaks or exits the loop.
    if ($index == 8)
        break;

}
```

This code snippet prints out:

10
9

When the index variable is equal to 8, break statement is executed and it results in breaking or exiting the loop. That way, programmers are able to exit a loop in whichever specific condition they want.

Continue

```php
$index = 10;
while($index > 0) {
    // Passes to next iteration immedately
    // for the values of 8 and 5.
    if ($index == 8 || $index == 5) {
        $index--;
        continue;
    }

    print("<pre>" . $index . "<br></pre>");
    $index--;
}
```

This code snippet prints out:

10
9
7
6
4
3
2
1

When the index variable is equal to 8 and 5, continue statement is executed and this statement passes to the next iteration or cycle immediately. After executing the continue statement, the rest of the code in the loop is not executed for the specified cycle and next cycle starts to execute immediately. That way, programmers are able to manipulate a loops functionality by passing to next cycles.

C

a) What are the location of tests in Logically Controlled Loops ? Pretest, posttest, or both?

In C, there are two Logically Controlled Loops which are **while** and **do-while** loops.

While Loops

```
int num = 10;
int preTestFlag = 1;

while( num < 10 ) {
    preTestFlag = 0;

    printf("While Loop in C is Post Test\n");
    num++;
}

if (preTestFlag)
    printf("While Loop in C is Pre Test\n");
```

This code snippet prints out:

While Loop in C is Pre Test

If C While Loop would have post test, then no matter what num is, it would execute the while loop at least once, hence, pretTestFlag variable would be false and the program would output Post Test condition which is stated inside the loop. Since C while loop has pre test, the check is done before the execution of the loop and because num is 10, the loop is not executed and it is stated that C while loop is pretest by the program.

As a conclusion, **C while loop has pre test.**

Do - while Loops

```c
int num2 = 10;
int preTestFlag2 = 1;

do {
    preTestFlag2 = 0;
    printf("Do-While Loop in C is Post Test\n");
    printf("Value in the loop is %d\n", num);
    num2 += 1;
} while (num < 10);

if (preTestFlag2)
    printf("Do-While Loop in C is Pre Test\n");
```

This code snippet prints out:

Do-While Loop in C is Post Test
Value in the loop is 10

If C Do-While Loop would have pre test, then the test or check would be done before the execution of the loop and since num is 10, the do-while loop would not execute. However, because C Do-While loop is post test, no matter what num is, it executes the while loop at least once. That's why, it prints out that C Do-While loop is post test and the value in the loop is 10.

As a conclusion, **C Do-While loop has post test.**

b) What kind of user-located loop control mechanisms are provided in C?

In C, there are two user-located loop control mechanisms which are **break** & **continue**.

Break

```c
int index = 10;
while (index > 0) {
    printf("%d\n", index);
    index -= 1;

    // Breaks or exits the loop.
    if (index == 8)
        break;
}
```

This code snippet prints out:

10
9

When the index variable is equal to 8, break statement is executed and it results in breaking or exiting the loop. That way, programmers are able to exit a loop in whichever specific condition they want.

Continue

```c
int index2 = 10;
while (index2 > 0) {
    // Passes to next iteration immedately
    // for the values of 8 and 5.
    if (index2 == 8 || index2 == 5) {
        index2 -= 1;
        continue;
    }

    printf("%d\n", index2);
    index2 -= 1;
}
```

This code snippet prints out:

10
9
7
6
4
3
2
1

When the index variable is equal to 8 and 5, continue statement is executed and this statement passes to the next iteration or cycle immediately. After executing the continue statement, the rest of the code in the loop is not executed for the specified cycle and next cycle starts to execute immediately. That way, programmers are able to manipulate a loops functionality by passing to next cycles.

Perl

a) What are the location of tests in Logically Controlled Loops ? Pretest, posttest, or both?

In Perl, there are two Logically Controlled Loops which are **while** and **do-while** loops.

While Loops

```
$num = 10;
$preTestFlag = 1;

while( $num < 10 ) {
    $main::preTestFlag = 0;
    print "While Loop in Perl is Post Test\n";
    $main::num++;
}

if ($preTestFlag) {
    print "While Loop in Perl is Pre Test\n";
}
```

This code snippet prints out:

While Loop in Perl is Pre Test

If Perl While Loop would have post test, then no matter what num is, it would execute the while loop at least once, hence, pretTestFlag variable would be false and the program would output Post Test condition which is stated inside the loop. Since Perl while loop has pre test, the check is done before the execution of the loop and because num is 10, the loop is not executed and it is stated that Perl while loop is pretest by the program.

As a conclusion, Perl **while loop has pre test.**

Do - while Loops

```
$num = 10;
$preTestFlag = 1;

do {
    $main::preTestFlag = 0;
    print "Do-While Loop in Perl is Post Test\n";
    print "Value in the loop is " . $num . "\n";
    $num++;
} while ($num < 10);

if ($preTestFlag) {
    print "Do-While Loop in Perl is Pre Test\n";
}
```

This code snippet prints out:

Do-While Loop in Perl is Post Test
Value in the loop is 10

If Perl Do-While Loop would have pre test, then the test or check would be done before the execution of the loop and since num is 10, the do-while loop would not execute. However, because Perl Do-While loop is post test, no matter what num is, it executes the while loop at least once. That's why, it prints out that Perl Do-While loop is post test and the value in the loop is 10.

As a conclusion, **Perl Do-While loop has post test.**

b) What kind of user-located loop control mechanisms are provided in Perl?

In Perl, there are two user-located loop control mechanisms which are **break** & **continue**.

Break

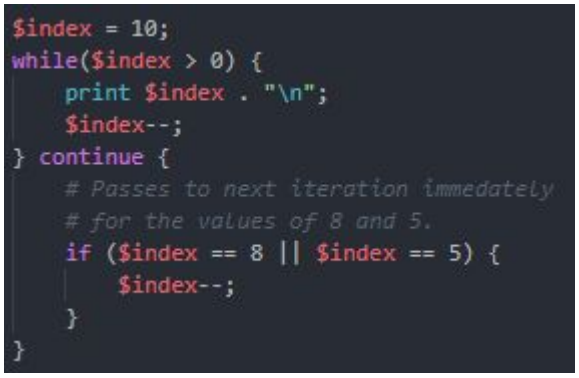
```
$index = 10;
while ($index > 0) {
    print "" . $index . "\n";
    $index--;

    # Breaks or exits the loop.
    if ($index == 8) {
        last;
    }
}
```

This code snippet prints out:

10
9

When the index variable is equal to 8, break statement is executed and it results in breaking or exiting the loop. That way, programmers are able to exit a loop in whichever specific condition they want.

Continue



```
$index = 10;
while($index > 0) {
    print $index . "\n";
    $index--;
} continue {
    # Passes to next iteration immedately
    # for the values of 8 and 5.
    if ($index == 8 || $index == 5) {
        $index--;
    }
}
```

This code snippet prints out:

10
9
7
6
4
3
2
1

When the index variable is equal to 8 and 5, continue statement is executed and this statement passes to the next iteration or cycle immediately. After executing the continue statement, the rest of the code in the loop is not executed for the specified cycle and next cycle starts to execute immediately. That way, programmers are able to manipulate a loops functionality by passing to next cycles.

Python

a) What are the location of tests in Logically Controlled Loops ? Pretest, posttest, or both?

In Python, there is one Logically Controlled Loop which is the **while** loop.

While Loops

```
num = 10
pretestFlag = True

while( num < 10 ):
    pretestFlag = False
    print("While Loop in Python is Post Test")
    num += 1

if (pretestFlag):
    print("While Loop in Python is Pre Test")
```

This code snippet prints out:

While Loop in Python is Pre Test

If Python While Loop would have post test, then no matter what num is, it would execute the while loop at least once, hence, pretTestFlag variable would be false and the program would output Post Test condition which is stated inside the loop. Since Python while loop has pre test, the check is done before the execution of the loop and because num is 10, the loop is not executed and it is stated that Python while loop is pretest by the program.

As a conclusion, **Python while loop has pre test.**

b) What kind of user-located loop control mechanisms are provided in Python ?

In Python, there are two user-located loop control mechanisms which are **break** & **continue**.

Break

```
index = 10
while (index > 0):
    print(index)
    index -= 1

    # Breaks or exits the loop.
    if index == 8:
        break
```

This code snippet prints out:

10
9

When the index variable is equal to 8, break statement is executed and it results in breaking or exiting the loop. That way, programmers are able to exit a loop in whichever specific condition they want.

Continue

```
index = 10
while (index > 0):
    # Passes to next iteration immedately
    # for the values of 8 and 5.
    if index == 8 or index == 5:
        index -= 1
        continue

    print(index)
    index -= 1
```

This code snippet prints out:

10
9
7
6
4
3
2
1

When the index variable is equal to 8 and 5, continue statement is executed and this statement passes to the next iteration or cycle immediately. After executing the continue statement, the rest of the code in the loop is not executed for the specified cycle and next cycle starts to execute immediately. That way, programmers are able to manipulate a loops functionality by passing to next cycles.