Name: Karan R Fulare email: fularekaran@yahoo.com

Assignment 7.1 (20 Pts)

Demonstrate following concepts using any programming language –

- For, While, Continue, If, Else, Switch
- Recursion
- Binary Search Tree
- Closure

1.2 - (20 pts)
Write an implementation of getElementById, which performs the same basic task as that of actual getElementById, (*don't use shortcuts like queryselector*)

8.2 (20pts)

Given a number x , find out if it is a prime number or not , use javascript and find out the difference between Next prime number after X and X

Solutions:

Assignment 7.1

- For, While, Continue, Else, Switch

1.For loop in java

For loop is used to loop/iterate over a value or data-set or to execute some code block, it runs until a give condition becomes false

It usually takes about three parameters first one(initialization) is an integer with some assigned value usually "0" next one(testing) is a condition when this condition becomes false loop terminates and last one(updating) is increment or decrement of the integer value. These parameters are separated by ";".

Eg.

```
for( int k=0 ; k<3; k++) {
System.out.println("karan");
}
```

The above loop has three parameters first one integer "k" with value as zero , the second parameter is used to terminate the loop , currently the value of k is less than three so it returns true and loop runs , when k will become greater than three the parameter will return false and the loop won't run ,next parameter is "k++" we are incrementing the value of "k" using post increment here , this will increment the k by one after the loop executes . all the three parameters are optional ,but that won't give us the right output.

Operations of loop :-

1. It will take value of k as zero then check the condition is k less than 3 ,ans is yes it is so it runs the code inside parenthesis which prints "karan" after every line within the parenthesis is executed the loop increments the value of k by one (can increase or decrease our choice here in this example we are increasing by one) .

2. Now the value of k is 1 , it checks the condition k<3 yes it is so runs the loop ,prints “karan” increments k by one .
3. Again does the same ,now notice k=2,after incrementing it will become 3 .
4. Now k=3 which does not satisfy the condition k<3 so it terminates the loop and does not execute the code .

For loop over a data-set (a array here)

Eg.

```
For(int i=0;i<=array.lenght;i++){
System.out.println(array[i]);
}
```

This loop prints all the elements of the array here our variable “i” works as a indexing variable.

2.while loop in java .

This loop is similar to for loop , the only difference is that here we provide only the testing statement

```
while(somethings true){
Do this;
}
```

The initialization can be done before the loop and the updating of the loop/control variable and be done at the last line of the loop .

Eg.

```
int k=0;        // initializing the control variable
while(k<3){       // initial test/checking
System.out.println(“karan” + k);
k++;       //updating my control variable
}
```

3.continue

This keyword is used to skip some statements, usually used with loops , when we write continue it exists the loop and goes for next iteration, directly .

Eg.

```
  for(int k=0;k<=5;k++) {
       if(k==3) {
          continue;        // when k=3 it will not execute the next lines the and go for k=4
             }
    System.out.println(k);                  // will not print 3
}
```

4.if else

If else statements are used for condition , the if part runs onlt if the condition satisfies , if it does not else part runs, simple . if the condition in if statement fails or is false the code in else statement always runs.

Eg.

```
 if(condition true){
        run this code}
else {
        run this code }
```

Eg.

```
if( 5 > 4 ){
        System.out.println("yes");
}
else {
        System.out.println("no");
}
```

5.Switch

if there are many if else statement we can use Switch case , here all the if are replaced by "case" and the else part can replaced by "default" if none of the cases runs default runs . you need to add "break" after each case so the switch terminates after case is found.

eg.

```
int month=2 ;
switch(month){
case 1:{ System.out.println("its Jan");
break; }
case 2:{ System.out.println("its Feb");
break; }
case 3:{ System.out.println("its March");
break; }
case 4:{ System.out.println("its April");
break; }
default :{ System.out.println("nothing"); }
}
```

5. Recursion

Recursion is when a function calls itself , recursion has three parts first one is the base condition which stops the recursive call and returns something or prints something , next part is the recursive call where we call the same function within , and the third part is some small calculations where we compute the smaller chunks . recursion works on smaller problems to make up the larger one .

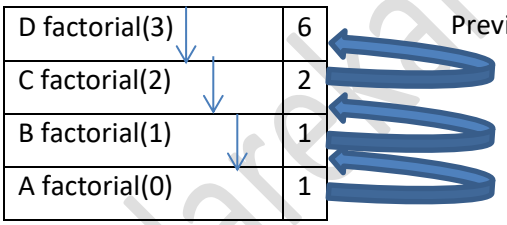
Eg.

```
1. public static int factorial(int n){
2.       If(n==0) {                  //base condition
3.                 return 1 ; }
4.       return n*factorial(n-1);     //small calculation and function call
   }
```

Here we are calling the factorial function within itself as we know that factorial of a number is n*n-1*n-2*n-3*…..; so here we provide that calculate first the factorial of n-1 and then multiply it by n , suppose n is 3 so factorial of 3 will be 3*(3-1)*(3-2)*(3-3) but for n=0 we are retuning 1 , our base case.

A. First call n=3, function will call itself with n-1 ie 2 here
B. Now n=2 , function calls itself with n-1 ie 1 here
C. Now n=1 ,function calls itself with n-1 ie 0 here
D. Now n=0, so function returns 1.

As all of this was going in call stack and it works in LIFO (last in first out) our last call was D. Which returned 1 this is input to C and after this it was C in the stack which will return 1*1 = 1 this is input to B , B with input as 1 will return 2*1=2 now this 2 is input to A which first entered in the call stack and is sitting at bottom now with input as 2 and n=3 it will return 3*2=6 and that is the ans .



Previous small calulations output is the input to next call .

6.Binary Search Tree

Binary search tree is a tree where all the left nodes are smaller than the root and right nodes , means left nodes always contain smaller values than that of the root and right node . also each of the subtree is a BST.

I know about BST but at this moment can't remember the implementation if it would have been a online test I would have not been able to attempt this one, sorry for that and I don't want to cheat I will consider is as my fate if I'm rejected for this question but now I'll focus on trees also, what I recall is that binary trees uses recursion a lot for searching.

6. Closure

Closure can be defined as a function with its lexical scope .Consider a function which is enclosed inside another function now the inside function have access to the environment of the outer function too that's what we call as a closure.

Eg.

```
function x() {
      var a=7;
       function y() {
              console.log(a);
    }
    y();
}
x()
```

1.2 Solution

Here I'm using the document which hold the data and then I use tags to get my elements and then I'm checking the element with specified id if present it returns the element it return null same as described . below is the implementation.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
  </head>
  <body>
    <div id="open">
      <h1>Do your best then Don't worry Be happy - Jai Meher Baba</h1>
    </div>

    <script>
      function ourOwnGetElementById(id) {
        var doc = document;
        var elements  = doc.getElementsByTagName("*");
        for (var i = 0; i < elements.length; i++) {
          if (elements[i].id == id) {
            return elements[i];
          }
        }
        return null;
      }

      var a = ourOwnGetElementById("open");
      console.log(a);
    </script>
  </body>
</html>
```

8.2 solution

Simple solution it is .

```
function checkPrime(n) {
  if (n <= 1) {
    return false;
  } else if (n == 2) {
    return true;
  }
  for (var i = 2; i <= Math.sqrt(n); i++) {
    if (n % i == 0) {
      return false;
    }
  }
  return true;
}

console.log(checkPrime(5));

function findDifference(n) {
  let next = n + 1;
  while (!checkPrime(next)) {
    next++;
  }

  console.log(next + " this is the next prime no");
  return next - n;
}

console.log(findDifference(60) + " difference");
```