Sukkur IBA University
Department of Computer Science

DATA STRUCTURES
Lab06-Recursion

Instructor: Saif Hassan

READ IT FIRST

Prior to start solving the problems in this assignments, please give full concentration on following points.

1. WORKING – This is individual lab. If you are stuck in a problem contact your teacher, but, in mean time start doing next question (don't waste time).
2. DEADLINE – 11th March, 2022
3. SUBMISSION – This assignment needs to be submitted in a soft copy.
4. WHERE TO SUBMIT – Please visit your LMS.
5. WHAT TO SUBMIT – Submit this docx and pdf file.

KEEP IT WITH YOU!

1. Indent your code inside the classes and functions. It's a good practice!
2. It is not bad if you keep your code indented inside the loops, if and else blocks as well.
3. Comment your code, where it is necessary.
4. Read the entire question. Don't jump to the formula directly.

I, ___**Amjad Ali**___ with student ID _**191-21-0001**__

Section _**A**_hereby declare that I do understand the instructions above and follow them. This is my own work.

Exercises

Task1 Description

Task 00: (Simple Recursion)

a) Write a program to ask user input N and print numbers from 1 – N in ascending/descending order. (using recursion)
b) Print 1d character array values using recursion in forward/reverse direction.

Solution:

(A-Part)

```
1. package lab06;
2.
3. import static lab06.PrintingNodes.printNodesRecursive;
4.
5. public class Lab06 {
6.
7.     public static  void printNumbers(int n)
8.     {
9.         if(n<1)
10.          {
11.              return;
12.          }
13.          else{
14.
15.                  printNumbers(n-1);
16.                  System.out.println(n);
17.          }
18.
19.     }
20.     public static void main(String[] args) {
21.         // TODO code application logic here
22.         long start = System.nanoTime();
23.         printNumbers(10);
24.         long end = System.nanoTime();
25.         System.out.println("("+(end-start)+")NanoSeconds");
26.     }
27.
28. }
```

Sample Input:

```
long start = System.nanoTime();
printNumbers(10);
long end = System.nanoTime();
System.out.println("("+(end-start)+")NanoSeconds");
```

Sample Output

```
1
2
3
4
5
6
7
8
9
10
(405892)NanoSeconds
BUILD SUCCESSFUL (total time: 0 seconds)
```

(B-Part)

Solution:

```
1. package lab06;
2.
3. public class CharRecursion {
4.
5.     public static void printCharForward(char arr[], int a) {
6.         if (a == arr.length) {
7.             return;
8.         } else {
9.             System.out.println(arr[a]);
10.              printCharForward(arr, a + 1);
11.         }
12.
13.     }
14.
15.     public static void printCharBackward(char arr[], int a) {
16.         if (a == arr.length) {
17.             return;
18.         } else {
19.
20.             printCharBackward(arr, a + 1);
21.             System.out.println(arr[a]);
22.         }
23.
24.     }
```

```
25.
26.      public static void main(String[] args) {
27.          char[] arr = {'a', 'b', 'c', 'd', 'e', 'f', 'g'};
28.          int a = 0;
29.          long start = System.nanoTime();
30.          printCharForward(arr, a);
31.          long end = System.nanoTime();
32.          System.out.println("(" + (end - start) + ")NanoSeconds");
33.
34.      }
35.
36.  }
```

Sample Input:

```
char[] arr = {'a', 'b', 'c', 'd', 'e', 'f', 'g'};
int a = 0;
long start = System.nanoTime();
printCharForward(arr, a);
long end = System.nanoTime();
System.out.println("(" + (end - start) + ")NanoSeconds");
```

Sample Output

```
run:
a
b
c
d
e
f
g
(273795)NanoSeconds
BUILD SUCCESSFUL (total time: 0 seconds)
```

Task2 Description

Task 01: (Fibonacci Series)

a) Write a program to generate Fibonacci series till N. N is any user input. (Using iterative approach)
b) Write a program to generate Fibonacci series till N. N is any user input. (Using recursive approach)
c) Calculate and compare time, whether a or b takes less time **(using code).**

Solution:

```
1. package lab06;
2.
3. public class FibunacciRecursive {
4.
5.     public static void printFibonnaciIterattive(int n) {
6.         int a = 1, b = 1, c = 2;
7.         boolean cond = false;
8.         System.out.println(a);
9.         System.out.println(b);
10.          while (c <= n) {
11.
12.              System.out.println(c);
13.              a = b;
14.              b = c;
15.              c = a + b;
16.          }
17.      }
18.
19.      public static void printFibonnaciRecursive(int n, int a,
   int b) {
20.          int c = a + b;
21.          if (a == 1 && b == 1) {
22.              System.out.println(a);
23.              System.out.println(b);
24.          }
25.          if (c > n) {
26.              return;
27.          } else {
28.              System.out.println(c);
29.              a = b;
30.              b = c;
31.              printFibonnaciRecursive(n, a, b);
```

```
32.         }
33.     }
34.
35.     public static void main(String[] args) {
36.         // printFibonnaciIterattive(8);
37.         printFibonnaciRecursive(8, 1, 1);
38.     }
39. }
```

Sample Input:

A-part(Recursive)

```
long start = System.nanoTime();
printFibonnaciRecursive(8, 1, 1);
long end = System.nanoTime();
System.out.println("(" + (end - start) + ")NanoSeconds");
```

B-Part(Ittterative)

```
long start = System.nanoTime();
printFibonnaciIterattive(8);
long end = System.nanoTime();
System.out.println("(" + (end - start) + ")NanoSeconds");
```

Sample Output

A-Part(Recursive)

```
1
1
2
3
5
8
(292004)NanoSeconds
BUILD SUCCESSFUL (total time: 0 seconds)
```

B-Part(Itterative)

```
run:
1
1
2
3
5
8
(303592)NanoSeconds
BUILD SUCCESSFUL (total time: 0 seconds)
```

>=Recursive>Iterative=<

Task3 Description

Task 02: (Factorial)

a) Design a method to calculate factorial of N number where N is any user input. (Using iterative approach)
b) Design a method to calculate factorial of N number where N is any user input. (Using recursive approach)
c) Calculate and compare time, whether a or b takes less time **(using code)**.

Solution:

```
1. package lab06;
2.
3. public class Factorial {
4.
5.     public static int findFactorialRecursive(int n) {
6.
7.         if (n == 0 || n == 1) {
8.             return 1;
9.         }
10.          return n * findFactorialRecursive(n - 1);
11.     }
12.
13.     public static int findFactorialIterattive(int n) {
14.         int result = 1;
15.         for (int i = 1; i <= n; i++) {
16.             result = result * i;
17.         }
18.
19.         return result;
20.     }
21.
22.     public static void main(String[] args) {
23.
   System.out.println(findFactorialIterattive(5));
24.     }
```

```
25. }
```

Sample Input:

A-Part (Recursive)

```
long start = System.nanoTime();
findFactorialRecursive(5);
long end = System.nanoTime();
System.out.println("(" + (end - start) + ")NanoSeconds");
```

B-Part (Iterative)

```
long start = System.nanoTime();
System.out.println(findFactorialIterattive(5));
long end = System.nanoTime();
System.out.println("(" + (end - start) + ")NanoSeconds");
```

Sample Output

A-Part (Recursive)

```
120
(184407)NanoSeconds
BUILD SUCCESSFUL (total time: 0 seconds)
```

B-Part (Iterative)

```
120
(189703)NanoSeconds
BUILD SUCCESSFUL (total time: 0 seconds)
```

>=Recursive>Iterative=<

Task4 Description

Task 03: (Printing Linkedlist):

a) Write a program to print all nodes from linkedlist. (Using iterative approach)
b) Write a program to print all nodes from linkedlist. (Using recursive approach)
c) Calculate and compare time, whether a or b takes less time **(using code).**

Solution:

```
1. package lab06;
2.
3. public class PrintingNodes {
4.
5.     public static void printNodesIterattive(Node head) {
6.         Node current = head;
7.         while (current != null) {
8.             System.out.println(current.name);
9.             current = current.next;
10.         }
11.     }
12.
13.     public static void printNodesRecursive(Node head) {
14.         if (head == null) {
15.             return;
16.         } else {
17.             System.out.println(head.name);
18.             printNodesRecursive(head.next);
19.         }
20.     }
21.
22.     public static void main(String[] args) {
23.         DoubleLinkedList list = new DoubleLinkedList();
24.         list.insertAtBeginning("Amjad");
25.         list.insertAtBeginning("Ahsan");
26.         list.insertAtBeginning("Sattar");
27.         list.insertAtBeginning("Fazal");
28.         list.insertAtBeginning("Khuraim");
29.         list.insertAtBeginning("Hamza");
30.         list.insertAtBeginning("Faraz");
31.         list.insertAtBeginning("Razaque");
32.         long start = System.nanoTime();
```

```
33.          printNodesRecursive(list.head);
34.          long end = System.nanoTime();
35.          System.out.println("(" + (end - start) +
   ")NanoSeconds");
36.
37.      }
38.
39.  }
```

Sample Input:

A-Part (Recursive)

```
long start = System.nanoTime();
printNodesRecursive(list.head);
long end = System.nanoTime();
System.out.println("(" + (end - start) + ")NanoSeconds");
```

B-Part (Iterative)

```
long start = System.nanoTime();
printNodesIterattive(list.head);
long end = System.nanoTime();
System.out.println("(" + (end - start) + ")NanoSeconds");
```

Sample Output

A-Part (Recursive)

```
Razaque
Faraz
Hamza
Khuraim
Fazal
Sattar
Ahsan
Amjad
(330740)NanoSeconds
BUILD SUCCESSFUL (total time: 0 seconds)
```

B-Part (Iterative)

```
Razaque
Faraz
Hamza
Khuraim
Fazal
Sattar
Ahsan
Amjad
(296639)NanoSeconds
BUILD SUCCESSFUL (total time: 0 seconds)
```

>= Iterative > Recursive =<