|  |  |  |
| --- | --- | --- |
| Sukkur_IBA_New_Logo | **Sukkur IBA University**  **Department of Computer Science** | **C:\Users\Saif Hassan\Downloads\CS logo (3).jpg** |

**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.

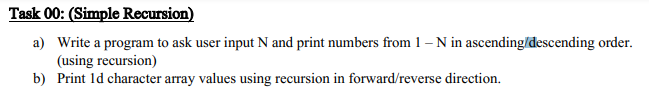
I, \_\_\_**Amjad Ali**\_\_\_ with student ID \_**191-21-0001**\_\_

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

my own work.

**Exercises**

**Task1 Description**

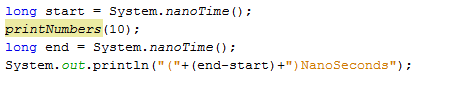


Solution:

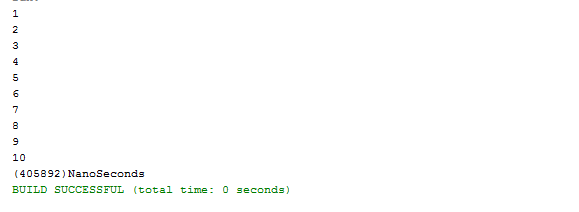
**(A-Part)**

1. package lab06;
3. import static lab06.PrintingNodes.printNodesRecursive;
5. public class Lab06 {
7. public static void printNumbers(int n)
8. {
9. if(n<1)
10. {
11. return;
12. }
13. else{
15. printNumbers(n-1);
16. System.out.println(n);
17. }
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. }
28. }

**Sample Input:**



**Sample Output**

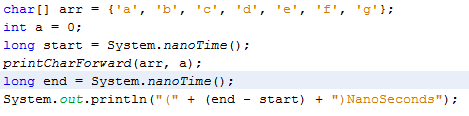


**(B-Part)**

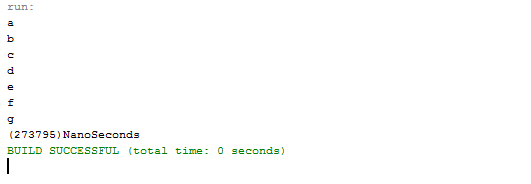
Solution:

1. package lab06;
3. public class CharRecursion {
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. }
13. }
15. public static void printCharBackward(char arr[], int a) {
16. if (a == arr.length) {
17. return;
18. } else {
20. printCharBackward(arr, a + 1);
21. System.out.println(arr[a]);
22. }
24. }
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");
34. }
36. }

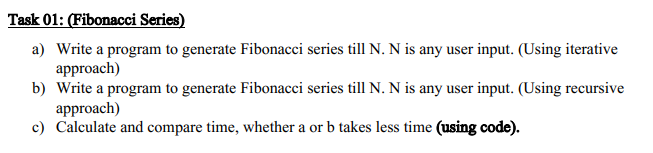
**Sample Input:**



**Sample Output**



**Task2 Description**

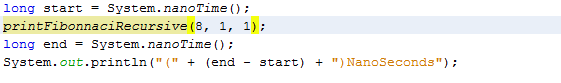


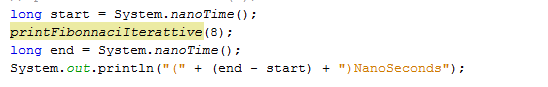
Solution:

1. package lab06;
3. public class FibunacciRecursive {
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) {
12. System.out.println(c);
13. a = b;
14. b = c;
15. c = a + b;
16. }
17. }
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. }
35. public static void main(String[] args) {
36. // printFibonnaciIterattive(8);
37. printFibonnaciRecursive(8, 1, 1);
38. }
39. }

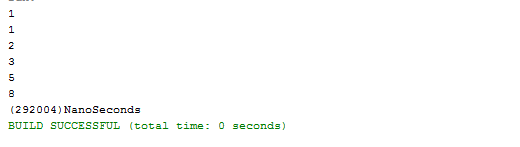
**Sample Input:**

A-part(Recursive)

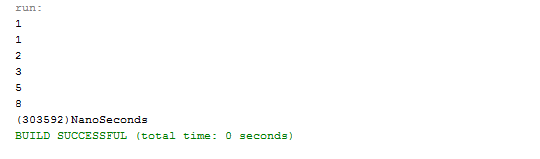
 B-Part(Ittterative)

**Sample Output**

A-Part(Recursive)

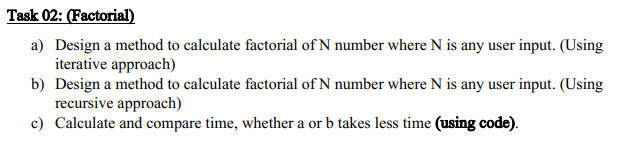


B-Part(Itterative)



**>=Recursive>Iterative=<**

**Task3 Description**

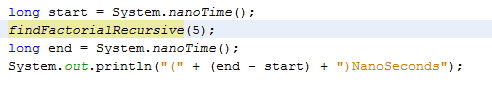


Solution:

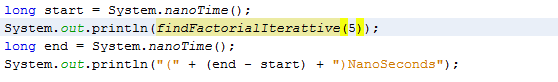
1. package lab06;
3. public class Factorial {
5. public static int findFactorialRecursive(int n) {
7. if (n == 0 || n == 1) {
8. return 1;
9. }
10. return n \* findFactorialRecursive(n - 1);
11. }
13. public static int findFactorialIterattive(int n) {
14. int result = 1;
15. for (int i = 1; i <= n; i++) {
16. result = result \* i;
17. }
19. return result;
20. }
22. public static void main(String[] args) {
23. System.out.println(findFactorialIterattive(5));
24. }
25. }

**Sample Input:**

A-Part (Recursive)



B-Part (Iterative)

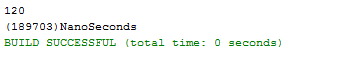


**Sample Output**

A-Part (Recursive)

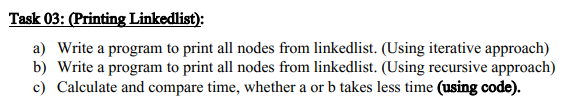


B-Part (Iterative)



**>=Recursive>Iterative=<**

**Task4 Description**

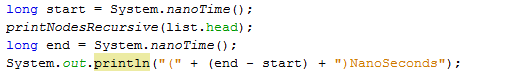


Solution:

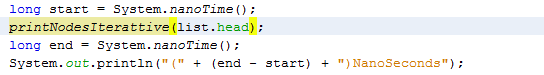
1. package lab06;
3. public class PrintingNodes {
5. public static void printNodesIterattive(Node head) {
6. Node current = head;
7. while (current != null) {
8. System.out.println(current.name);
9. current = current.next;
10. }
11. }
13. public static void printNodesRecursive(Node head) {
14. if (head == null) {
15. return;
16. } else {
17. System.out.println(head.name);
18. printNodesRecursive(head.next);
19. }
20. }
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");
37. }
39. }

**Sample Input:**

A-Part (Recursive)

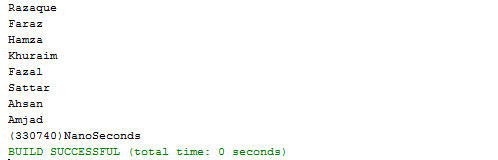


B-Part (Iterative)

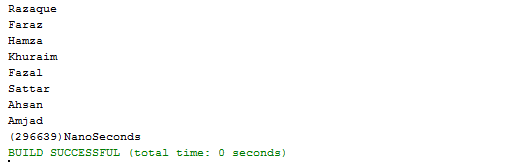


**Sample Output**

A-Part (Recursive)



B-Part (Iterative)



**>= Iterative > Recursive =<**