**Name: MUHAMMAD TAHA**

**CMS: 023-23-0377**

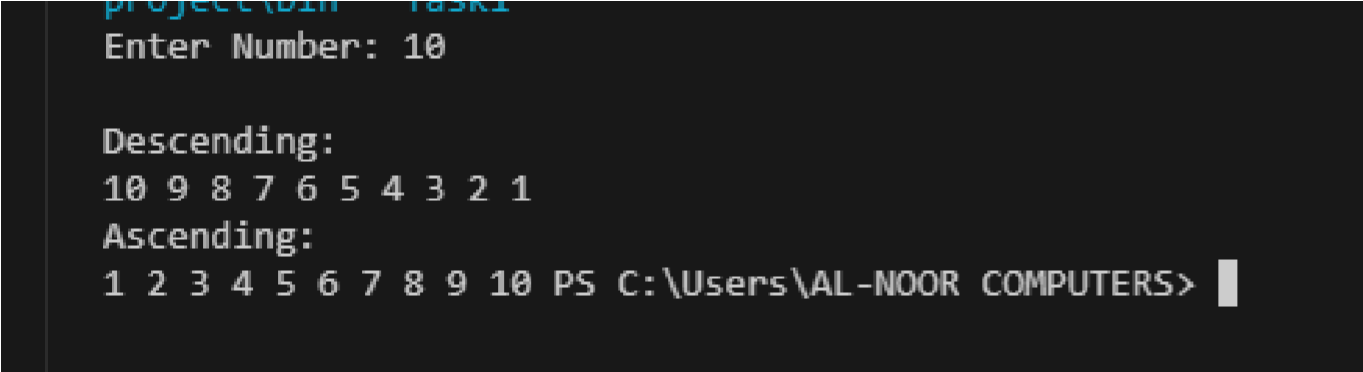
**Section: B**

**Lab: 07**

TASK 01

|  |
| --- |
| import java.util.Scanner; public class Task1 { void printNumbers(int n){ if (n > 0) { if (n==10) {  System.out.println();  System.out.println("Descending: ");  }  System.out.print(n + " "); printNumbers(n-1); if (n==1) {  System.out.println();  System.out.println("Ascending: ");  }  System.out.print(n + " ");  } else { return;  }  }  public static void main(String[] args) { Scanner sn = new Scanner(System.in);  Task1 t1 = new Task1();  System.out.print("Enter Number: "); int no = sn.nextInt(); t1.printNumbers(no);  }  } |

OUTPUT



TASK 02

|  |
| --- |
| public class Task2 {  public void CharDesc(char Arr[], int index){  if(Arr.length==index){ return;  }else{    CharDesc(Arr,index+1);  System.out.print(Arr[index] + " ");    }    } public void CharAsc(char Arr[], int index){  if(Arr.length==index){ return;  }else{    System.out.print(Arr[index] + " ");  CharAsc(Arr,index+1);  } }  public static void main(String[] args) {  Task2 t2 = new Task2();  char Arr[]={'s','h','a','k','o','o',’r’}; |
| System.out.println("Ascending:"); t2.CharAsc(Arr,0);  System.out.println();    System.out.println("Descending:"); t2.CharDesc(Arr, 0);  System.out.println();  }  } |

OUTPUT

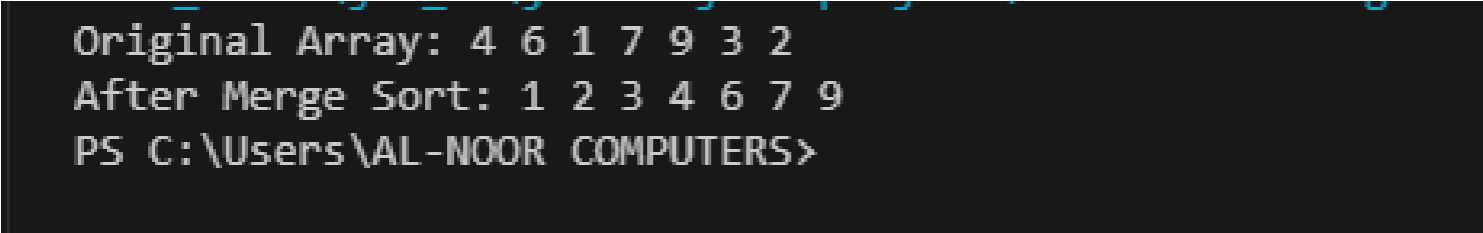
TASK 03

|  |
| --- |
| public class Task3MergeSort {  void Split(int arr[], int l, int r) {    if (l < r) {  int mid = (l + r) / 2;  Split(arr, l, mid);    Split(arr, mid + 1, r);    Merge(arr, l, mid, r); |
| }  }  void Merge(int arr[], int l, int mid, int r) {  int n1 = mid - l + 1; int n2 = r - mid;    int left[] = new int[n1]; int right[] = new int[n2];    for (int i = 0; i < n1; i++) { left[i] = arr[l + i];  } for (int j = 0; j < n2; j++) { right[j] = arr[mid + 1 + j];  }    int i = 0, j = 0; int k = l;    while (i < n1 && j < n2) { if (left[i] <= right[j]) { arr[k] = left[i]; i++; } else { arr[k] = right[j]; j++; } k++;  }    while (i < n1) { arr[k] = left[i]; i++; k++;  }  while (j < n2) { |

arr[k] = right[j];

|  |
| --- |
| j++; k++;  }  }  void display(int arr[]) {  for (int i = 0; i < arr.length; i++) {  System.out.print(arr[i] + " ");  }  }  public static void main(String[] args) {  Task3MergeSort ms = new Task3MergeSort();    int arr[] = {4,6,1,7,9,3,2};  System.out.print("Original Array: "); ms.display(arr);  System.out.println();    ms.Split(arr, 0, arr.length - 1);  System.out.print("After Merge Sort: "); ms.display(arr);  System.out.println();  }  } |

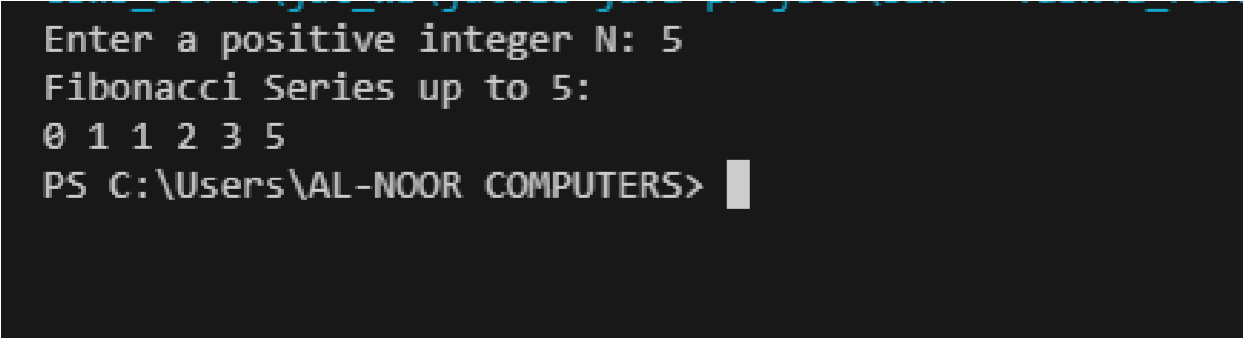
OUTPUT



TASK 04-A

|  |
| --- |
| import java.util.Scanner;  public class Task4a\_Fabonacci { // Through iterative public void generateFibonacci(int N) { int a = 0, b = 1;      while (a <= N) {  System.out.print(a + " "); int next = a + b; a = b; b = next;  }    System.out.println();  }  public static void main(String[] args) {  Scanner scanner = new Scanner(System.in);  Task4a\_Fabonacci t4 = new Task4a\_Fabonacci(); System.out.print("Enter a positive integer N: "); int N = scanner.nextInt();    System.out.println("Fibonacci Series up to " + N + ":"); t4.generateFibonacci(N);    scanner.close();  }    } |

OUTPUT



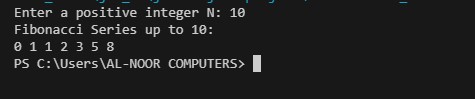
Task 04-B

|  |
| --- |
| import java.util.Scanner;  public class Task4b\_Fabonacci { // Through Recursion public int fibonacci(int n) { if (n <= 1) { return n;  } return fibonacci(n - 1) + fibonacci(n - 2);  }  public static void main(String[] args) {  Scanner scanner = new Scanner(System.in);  Task4b\_Fabonacci t4 = new Task4b\_Fabonacci();    System.out.print("Enter a positive integer N: "); int N = scanner.nextInt();    System.out.println("Fibonacci Series up to " + N + ":");    for (int i = 0; i <= N; i++) { int fib = t4.fibonacci(i); if (fib > N) { |

break;

|  |
| --- |
| }  System.out.print(fib + " ");  }    System.out.println(); scanner.close();  }    } |

## OUTPUT



**Task 05**

|  |
| --- |
| import java.util.\*; class Task5\_Factorail{ static double startTime, endTime, startTime1, endTime1;  public static int factorial(int n){ startTime1 = System.nanoTime(); //record the starting time if (n==1) { return 1;  }  return n\*factorial(n-1);    }    public static void main(String[] args) {  Scanner sc=new Scanner(System.in);  System.out.println();  System.out.println("Iterative Way"); |

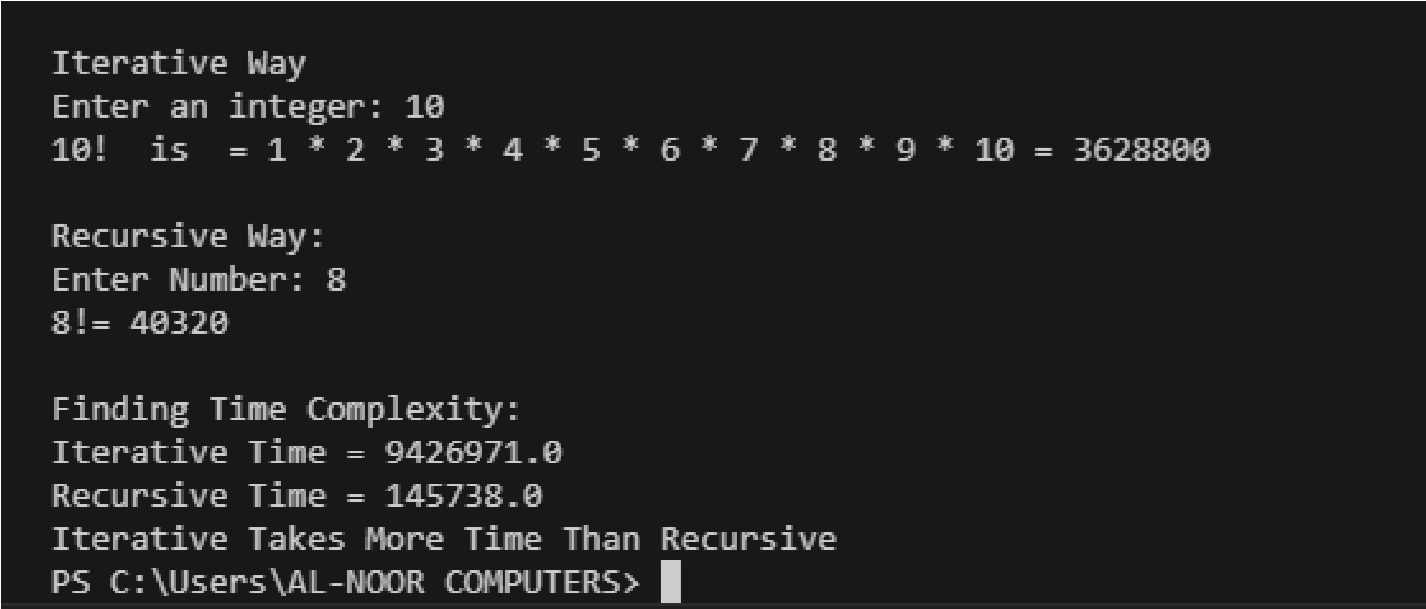
System.out.print("Enter an integer: "); int n=sc.nextInt(); // iterative approach int sum=1;

System.out.print(n+"!"+" is = "); for (int i = 1; i <=n; i++) { startTime = System.nanoTime(); //record the starting time

|  |
| --- |
| sum\*=i;  System.out.print(i);    if (i!=n)  System.out.print(" \* ");  } endTime = System.nanoTime(); //record the ending time double totalTime = endTime - startTime;  System.out.print(" = "+sum);  System.out.println();  System.out.println();  System.out.println("Recursive Way: ");  System.out.print("Enter Number: ");    int n1=sc.nextInt();    System.out.println(n1+"!= "+factorial(n1)); endTime1 = System.nanoTime(); //record the ending time double totalTime1 = endTime1 - startTime1;  System.out.println();  System.out.println("Finding Time Complexity:");  System.out.println("Iterative Time = " + totalTime1);  System.out.println("Recursive Time = " + totalTime);    if (totalTime<totalTime1) {  System.out.println("Iterative Takes More Time Than Recursive"); }else{  System.out.println("Recursive Takes More Time Than Iterative");  }          } |

}

# OUTPUT



**Task 06**

|  |
| --- |
| import java.util.Scanner;  class Task6\_print\_linkedlist{ int count=0; long startTime1,startTime; class Node{ int data;  Node next;  Node (int data)  {  this.data=data; this.next=null;    }  }    void printRecursive(Node n){ startTime1 = System.nanoTime(); //record the starting time if (n!=null) {  System.out.print(n.data+ " "); printRecursive(n = n.next);  }    } |

// add first Node head; public void addFirst(int data)

{ count++;

Node newNode =new Node(data); if (head==null) {

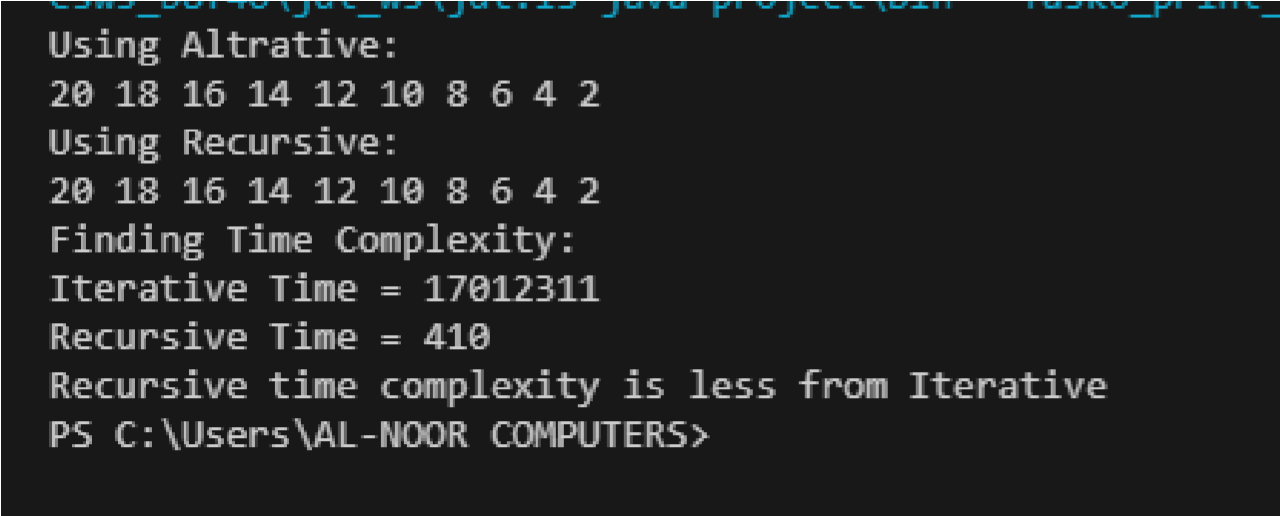
|  |
| --- |
| head=newNode;  }  else { Node temp; temp=head; head=newNode; newNode.next=temp;  } }  public static void main(String[] args) {  Scanner sc=new Scanner(System.in);  Task6\_print\_linkedlist ll = new Task6\_print\_linkedlist();  System.out.println("Using Altrative: ");  for (int i = 0; i <10 ; i++) { ll.startTime = System.nanoTime(); //record the starting time ll.addFirst(i+2+i);  }  Node temp1= ll.head; while (temp1!=null) {  System.out.print(temp1.data+" "); temp1=temp1.next;    }  long endTime = System.nanoTime(); //record the ending time long totalTime = endTime -ll.startTime;//compute the elapsed time  System.out.println();  System.out.println("Using Recursive: ");  Node tp = ll.head;    ll.printRecursive(tp); long endTime1 = System.nanoTime(); //record the ending time long totalTime1 = endTime1 - ll.startTime1;//compute the elapsed time  System.out.println(); |

System.out.println("Finding Time Complexity:");

System.out.println("Iterative Time = " + totalTime);

|  |
| --- |
| System.out.println("Recursive Time = " + totalTime1); if (totalTime<totalTime1) {  System.out.println("Iterative time complexity is less from Recursive");    }else{  System.out.println("Recursive time complexity is less from Iterative");  }      }    } |

### OUTPUT



**Task 07**

|  |
| --- |
| import java.util.Random; import java.util.Scanner; public class Search\_Using\_Recursive { static long startTime1,startTime;  static int srchElement(int arr[], int n){ startTime1 = System.nanoTime(); for (int i = 0; i < arr.length; i++) { if (arr[i]==n) { return 1; |

}

}

return 0;

} // recursion static int i=9; static int findElements(int arr[],int n){

|  |
| --- |
| startTime = System.nanoTime(); if (i>=0) {  if (arr[i]==n) { return 1;  }else{ i--; findElements(arr, n);  }    }  return 0; } public static void main(String[] args) { int arr[]=new int[10];  Random ra=new Random();  Scanner sc=new Scanner(System.in);  System.out.println("In Itration way"); System.out.println("Enter Number: "); int srch=sc.nextInt();    for (int i = 0; i < 10; i++) {    arr[i]=ra.nextInt()%100;    }    System.out.println( srchElement(arr, srch)); long endTime1 = System.nanoTime(); long totalTime1 = endTime1 - startTime1; if (srchElement(arr, srch)==0) {  System.out.println("Element not found");  }else{  System.out.println("Element found"); |

}

|  |
| --- |
| System.out.println("In Recursive Way");    System.out.println(srch);  System.out.println( findElements(arr, srch)); long endTime = System.nanoTime(); long totalTime = endTime - startTime; if (findElements(arr, srch)==0) {  System.out.println("Element not found");  }else{  System.out.println("Element found");    }    System.out.println("Finding out the time complexity ");  System.out.println("Iterative time Complexity: "+totalTime1);  System.out.println("Recursive time Complexity: "+totalTime);    if (totalTime1>totalTime) {  System.out.println("Iterative time complexity is greater than  Recursive.");  }else{  System.out.println("Recursive time complexity is greater than  Iterative.");    }            }  } |

## OUTPUT

