Recursion Prat 1

Lecture 1

* What is Recursion

Anwer:Definataion:The process in which a function calls itself directly or indirectly is called recursion.

Defination: Recursion is method of solving a computational problem where the solution depends on solution to smaller instance of same problem.

Defination: recursion is dividing into smaller problem instances of similar problems.

In the recursion we basically decrease the level of problem into one step or you want that level.

Q ) Calculate f(5) f->factorial?

Answer: using recursion f(5)=5\*f(4)

F(4)=4\*f(3)

F(3)=3\*F(2)

F(2)=2\*F(1)

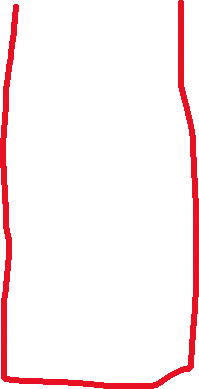
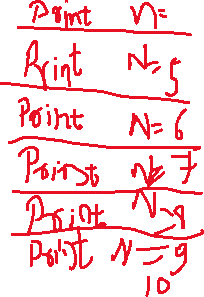
F(1)=1\*F(0)

F(0)=1

F(n)=n\*F(n)

Lecture No:02 Print number from n to 1(Decresing order)

public class DecresingNumber {  
  
 public static void printNumber(int x){   
 if(x==1){  
 System.*out*.println(x);  
 return;  
 }  
 System.*out*.println(x);  
 *printNumber*(x-1);  
  
 }  
 public static void main(String[] args){  
 *printNumber*(10);  
 }  
}



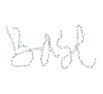
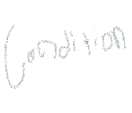
Lecture no 3 : stack overflow

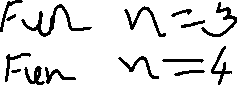
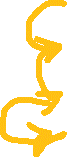
Lecture No 4: Print number in increasing order

public static void incersingOrderNumber(int n){  
 if(n==1){  
 System.*out*.println(n);  
 return;  
 }  
 *incersingOrderNumber*(n-1);  
 System.*out*.println(n);  
 }  
  
 public static void main(String[] args){  
 *incersingOrderNumber*(10);  
  
 }  
}









Base condition is hiited

From base condition it is starting p print the number.

Lecture No 5: Find factorial of n

public class FActorialNumber {  
 public static int factorialFun(int n){  
 if(n==1){  
 return 1;  
 }  
 int factorial=n\**factorialFun*(n-1);  
 return factorial;  
  
 }  
 public static void main(String[] args){  
 System.*out*.println(*factorialFun*(4));  
  
 }  
}



Fun n=1



Fun n=2



Fun n=3



Func n= 4



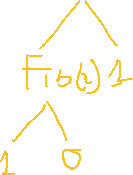
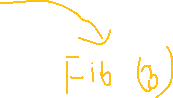
Lecture no:6 Print sum of first natural number

N+F(n+1)

public class SumOFNaturalNumber {  
 public static int sum(int n){  
 if(n==1){  
 return 1;  
 }  
 int sumOfNumber= n+*sum*(n-1);  
 return sumOfNumber;  
 }  
 public static void main(String[] args){  
 System.*out*.println(*sum*(3));  
 }  
  
}

Lecture No 7: Print Nth Fibonacci number

N^th Fibonacci number= previous two number sum



public class FibonaciSeries {  
 public static int fibonacciNumber(int n){  
 if (n==1||n==0){  
 return n;  
 }  
 int fibonumber=*fibonacciNumber*(n-1)+*fibonacciNumber*(n-2);  
 return fibonumber;  
  
 }  
 public static void main(String[] args){  
 System.*out*.println(*fibonacciNumber*(5));  
 }  
}

Lecture 8: Check if array is sorted or not

public class IsSorted {  
 public static boolean isSorted(int array[],int i){  
 if(i==array.length-1){  
 return true;  
 }  
 if(array[i]>array[i+1]){  
 return false;  
 }  
 return *isSorted*(array,i+1);  
  
 }  
 public static void main(String[] args){  
 int array[]={1,2,3,4,5,5,6};  
 int i=0;  
 System.*out*.println(*isSorted*(array,i));  
 int array2[]={1,2,3,4,50,5,6};  
 System.*out*.println(*isSorted*(array2,i));  
 }  
}

Lecture no 9: Find element from array if yes return the index if not return -1;

Given: Array[],Key, Startindex

public class FirstOcurrence {  
 public static int intergerOccurence(int array[],int key,int i){  
 if(array[i]==key){  
 return i;  
 }  
 if(i== array.length-1){  
 return -1;  
 }  
 return *intergerOccurence*(array,key,i+1);  
  
 }  
 public static void main(String[] args){  
 int array[]={23,4,55,67,78};  
 int i=0;  
 int key=7;  
 System.*out*.println(*intergerOccurence*(array,key,i));  
 }  
}

Lecture 10: Last Occurance number in array

Given: Array[],Key,lengthOfarray

Find the index of element which last occurance in array



Because 2 is key and it is last index is sorry 9

public class LAstOccurance {  
 public static int lastOccurance(int array[],int key, int lengthOfarray){  
 if(array[lengthOfarray]==key){  
 return lengthOfarray;  
 }  
 if(lengthOfarray==0&& array[lengthOfarray]!=key){  
 return -1;  
 }  
 return *lastOccurance*(array,key,lengthOfarray-1);  
 }  
 public static void main(String[] args){  
 int array[]={1,2,3,4,4,5,6,6,78,90};  
 int length=array.length-1;  
 int key=7;  
 System.*out*.println( *lastOccurance*(array,key,length));  
 }  
}

Lecture no 11: Power of X^n

Given x^n you have to calculate that

So what is approach to solve **x \* f(x, n-1);**

Suppose you want to calculate 2^5

So 2\***2\*2\*2\*2**

Like 2\*5=32

2\*4=16

2\*3=8

2\*2=4

2\*1=2

2\*0=1

Answer=32

public class PowerOfN {  
 public static int powerOfElement(int x,int n){  
 if(n==0){  
 return 1;  
 }  
 int power=x\**powerOfElement*(x,n-1);  
 return power;  
  
 }  
 public static void main(String[] args){  
 int x=2;  
 int n=5;  
 System.*out*.println(*powerOfElement*(x,n));  
 }  
}

Lecture 12: Power of N with optimized approach

public class PowerOfOptimisedApproch {  
 public static int powerOfelement(int x, int n){  
 if(n==0){  
 return 1;  
 }  
  
 int half= *powerOfelement*(x, n / 2);  
 int halfSqure=half\*half;  
  
// odd number  
 if(n/2!=0) {  
 int oddNumber = x\*halfSqure\*halfSqure;  
 return oddNumber;  
 }  
 return halfSqure;  
  
 }  
 public static void main(String[] args){  
 int x=2;  
 int n=5;  
 System.*out*.println(*powerOfelement*(x,n));  
}}

Approch: If n is even

Then

If(n==0){

Return n;

}

Return ***fun(x,n/2)\*func(x,n/2);***

If n is odd then

Return ***x\*fun(x, n/2)\*func(x, n/2);***

------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**Live Session: Recursion Part 1**

**Q1)Find the sum of N Element**

* **N=5 5+4+3+2+1=15**

**Int Recursion\_Function(int n){**

**If(n==1){**



**Return 1;**



**}**

**Int sum= n+ Recursion\_Function(n-1);**

**Return sum;**

**}**

**Q2) For a given integer array of size N. You have to find all the occurrences(indices) of given Element(key) and Print Them, Use a recursive function to solve this problem.**

**Sample input: arr[]={3,2,4,5,2,5,6,8,2,8,2}, Key=2**

**Sample Output: 1 4 8 9**

**Answer:**

**1)base case if (array.length==0&& array[0]!=key){**

**Return[ -1];}**

**2)if(array[length]==key){**

**Answer array[i]=length;**

**i++;**

**}**

**Function(arr,key,n-1);**

**Q.3) You are given a number(e.g 2019),convert it into a string of English like “ two zero one nine”.use a recursive function to solve the problem.**

**Note: The digits of the number will only be in the range 0-9 and the last digit of a number can’t be 0;**

**Sample input: 1947**

**Sample Output: “One nine four seven”**

**Approch🡺 creating a string array[]={“zero”, “one”, “two” upto nine}**

**To getting last digit of number divding number to 10 like 1947/10 what will be remainder it will print the that digit index of array string letter.**

**Creating recursive call for ech function to calculate last digit of number**

**Code :**

**Static String digits[]={“Zero”, “One”, “Two”, “Three”, “Four”, “Five”, “Six”, “Seven”, “Eight”, “Nine”}**

**Public static void printDigits(int number){**

**If(number==0)**

**{**

**Return ;**

**}**

**Int lastDigit= number%10;**

**printDigit(number/10);**

**System.out.println(digits[lastDigit] + “ “);**

**}**