**DATA STRUCTURE & JAVA**

**1)Define Scanner S=new Scanner (system.in) briefly?**

In Java, we take input from user with the help of **Scanner** class. Java has a number of predefined classes which we can use.

Predefined classes are organized in the form of packages. This **Scanner** class is found in **java.util** package. So to use Scanner class, we first need to include java.util package in our program.

**import java.util.Scanner; // This will import just the Scanner class  
import java.util.\*; // This will import the entire java.util package**

After importing, we need to write the following statement in our program.

**Scanner s = new Scanner (System.in);**

Here, by writing **Scanner s**, we are declaring **s** as an object of **Scanner** class. **System.in**within the round brackets tells Java this will be System Input i.e. input will be given to the system.

Taking a value from user is quite easy. Consider the following code.

**int n;  
n = s.nextInt(); *//*s is object of Scanner class**

Here,statement **n = s.nextInt();** is used to**input value of an integer variable 'n' from user.** Here, **nextInt()** is a method of the object **s** of Scanner class.

Similarly, we can input values of other data types also. Same as nextInt() is used to input an integer value, methods to input values of other data types are listed below.

**SUM RULE**

The sum and product rules of probability refer to methods of figuring out the probability of two events, given the probabilities of each event. The sum rule is for finding the probability of either of two events that cannot occur simultaneously. The product rule is for finding the probability of both of two events that are independent.

## Explaining the Sum Rule

Write the sum rule and explain it in words. The sum rule is given by P(A + B) = P(A) + P(B). Explain that A and B are each events that could occur, but cannot occur at the same time.

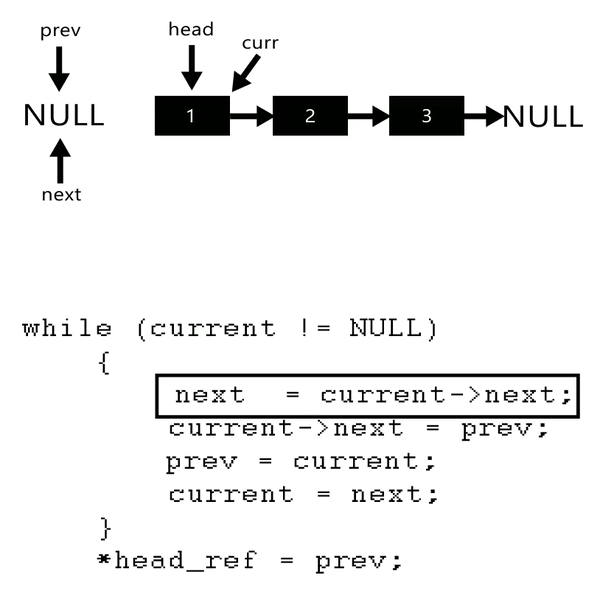
Give examples of events that cannot occur simultaneously and show how the rule works. One example: The probability that the next person walking into class will be a student and the probability that the next person will be a teacher. If the probability of the person being a student is 0.8 and the probability of the person being a teacher is 0.1, then the probability of the person being either a teacher or student is 0.8 + 0.1 = 0.9.

Give examples of events that can occur at the same time, and show how the rule fails. One example: The probability that the next flip of a coin is heads or that the next person walking into the class is a student. If the probability of heads is 0.5 and the probability of the next person being a student is 0.8, then the sum is 0.5 + 0.8 = 1.3; but probabilities must all be between 0 and 1.

## Product Rule

Write the rule and explain the meaning. The product rule is P( E\_F) = P(E)\_P(F) where E and F are events that are independent. Explain that independence means that one event occurring has no effect on the probability of the other event occurring.

**LINK LIST**



**EXPLANATION**

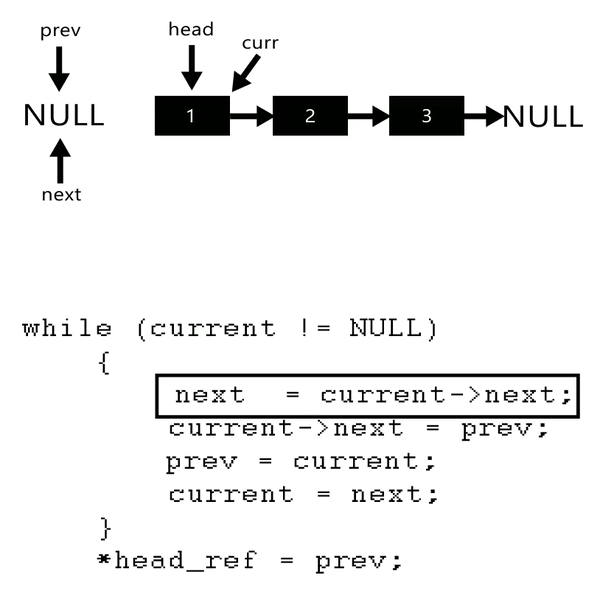
(This is the change and the function for making a link list print In reverse order)

**Step 1**:next will hold address current.next item,i.e. the next node just after current node

**Step 2**:value of current.next will be previous,i.e.the “next” address part of current.next will hold the prev address now.

**Step 3**:prev is now shifted to the value of current,and both pointing same node now.

**Step 4**:current now will store the next node leaving “prev” pointer behind.



Below

**HASHING**

HashMap is a part of java.util.package and it is for implementing hashing in java.  
Suppose I give you and array like - {1,2,1,3,4,5,6,7,8,8,1,1,1,9,9}  
and now I ask you the count of each element of the array, then what would be the naive approach?…to iterate through the array for each element and if that element is found then just increment the count for that element.  
However HashMap is used to establish a {value,key} relationship. i.e for each value you can allot a key. In this case since iterating through all the elements takes O(n\*n) complexity we can use HashMap to reduce the complexity to O(n). How??….the following program shows -

import java.util.\*;

class Hash{

public static void main (String[] args) {

int[] a={1,2,1,3,4,5,6,7,8,8,1,1,1,9,9};

HashMap<Integer,Integer> hm=new HashMap<Integer,Integer>();//HashMap instance created

for(int x: a){

if(hm.get(x)==null)hm.put(x,1);

else hm.put(x,hm.get(x)+1);

}

for(int x:a)System.out.println("The count of "+x+" is "+hm.get(x));

}

}

So what is happening in the above piece of code??…First I am taking an array which can be a user input array too, then I am using a HashMap object as “HashMap<Integer,Integer> hm=new HashMap<Integer,Integer>();”  
Here it means that the first parameter which is the key is going to be the integer and the second parameter is going to be the value which will store the count of the key in this case.  
Thus while iterating the array what i am doing is that first I am checking whether a particular value is present in the array or not using this line -” if(hm.get(x)==null)”, because by default the get() function of HashMap class returns null, and assigning 1 to the value for that key (i.e the particular value in the array for which we get null). This means this was the first encounter of that element and now its count is 1. Now if the get() function doesn’t returns null it means that the particular key has already been encountered so what i am doing is I am changing the value for this key to value+1 i.e I am incrementing the value for a key each time it is encountered (hm.put(x,hm.get(x)+1); the put\*+() function puts a particular value for the key here using the get() function I am getting the previous value or count of that key and putting back hm.get(x)+1 i.e previous count of x + 1 which just made the value as 2 which is now the current count of element x ).

Thus in this way using HashMap we can do this in O(n) time. We can use array for the process too but what if the numberss are in long range. Arrays don’t support indexes of long range. In that case we can use HashMap as HashMap<Long,Integer> hm=new HashMap<Long,Integer>()…and then the same process.

Not only Integer or Long or any predefined wrapper class but HashMap can also make a {key,value} relationship for any object whose class is defined in by the user.

**TIME COMPLEXITY CHART:**