```
        System.out.println(amt);
    }
}
```

Question 11.

A bookshelf is designed to store the books in a stock with LIFO (Last in First Out) operation. Define a class Book with the following specifications :

Class Name	:	***Book***
Data members/instance variables:		
name [ ]	:	*stores the names of the books*
point	:	*stores the index of the topmost book*
	:	*stores the maximum capacity of the bookshelf*
Methods/Member functions :		
Book (int cap)	:	*constructor to initialize the data members = cap and point = – 1*
void tell ( )	:	*displays the name of the book which was last entered in the shelf. If there is no book left in the shelf, displays the message SHELF EMPTY'*
void add (string v)		*adds the name of the book to the shelf it possible, otherwise displays the message "SHELF FULL"*
void display ( )	:	*displays all the names of the books available in the shelf*

*Specify the class **Book** giving the details of **ONLY** the functions **void tell ( )** and **void add (String)**. Assume that the other functions have been defined.*

The main function need not be written.

Answer 11.

```
class Book
{
  String name[];
  int point,max;
  void tell()
  {
    if(point== -1)
    {
System.out.println("SHELF EMPTY");
    }
    else
    {
System.out.println(name[point]);
    }
  }
  void add(String s)
  {
    if(point==max-1)
    {
System.out.println("SHELF FULL");
    }
    else
    {
      name[++point]=s;
    }
  }
}
```

Question 12.

(a) *A linked list is formed from the objects of the class Node. The class structure of the Node is given below :* **[2]**

```
class Node
{
        String name;
        Node next :
}
```

specified.

(Flowcharts are **not** required)

Question 11.

WordPile is an entity which can hold maximum of 20 characters. The restriction is that a character can be added or removed from one end only.

Some of the members of classes are given below:

Class Name	:	**WordPile**
Data members/instance variables:		
ch[ ]	:	character array to hold the character elements
capacity	:	integer variable to store the maximum capacity
top	:	to point to the index of the topmost element
Method/ Member functions:		
WordPile (int cap)	:	constructor to initialize the data member capactiy = cap, top = -1 and create the WordPile
void pushChar(char v)	:	adds the character to the top of WordPile if possible, otherwise output a message "WordPile is full"
char popChar()	:	returns the deleted character from the top of the WordPile if possible, otherwise it returns '\'

(a) *Specify the class **WordPile** giving details of the **constructor**, **void pushChar(char)** and **Char popChar( )**. **The main function and algorithm need not be written.*** **[8]**

(b) *What is the name of the entity described above and state one of its applications.* **[2]**

Answer 11. (a)

```
class WordPile
{
  char ch[];
  int capacity,top;
  public WordPile(int cap)
  {
    capacity=cap;
    top=-1;
    ch=new char[capacity];
  }
  public WordPile()
  {
    capacity=20;
    top=-1;
    ch=new char[capacity];
  }
  void pushChar(char v)
  {
    if(top==capacity-1)
    {
    System.out.println("WordPile is
    Full");
    }
    else
    {
      ch[++top]= v;
```

Question 12.

A stack is a linear data structure which enables the user to add remove integers from one end only, using the concept of LIFO (Last in First Out). An array containing the marks of 50 students in ascending order is to be pushed into the stack.

Define a class ***Array_to_Stack*** *with the following details :* **[10]**

Class name **:** ***Array_to_stack***

Data members/ instance variables **:**

m[ ]	:	*to store the marks*
st[ ]	:	*to store the stack elements*
cap	:	*maximum capacity of the array and stack*
top	:	*to point the index of the topmost element of the stack*

Methods/Member functions **:**

Array_to_Stack (int n)	:	*parameterized constructor to initialize cap = n and top = – 1*
void input_marks()	:	*to input the marks from user and store it in the array m [ ] in ascending order and simultaneously push the marks into the stack st[] by invoking the function pushmarks().*
void pushmarks (int v)	:	*to push the marks into the stack at top location if possible, otherwise, display "not possible"*
int popmarks( )	:	*to return marks from the stack if possible, otherwise, return-999*
void display()	:	*To display the stack elements.*

Specify the class ***Array_to_Stack****, giving the details of the* ***constructor(int)****,* ***void intput_marks()****,* ***void pushmarks(int)****,* ***int popmarks()*** *and* ***void display()****.*
The main function and the algorithm need not be written.

Answer 12.

```
import java.io.*;
class Array_to_Stack
{
  int m[],st[],cap,top;
  public Array_to_Stack(int n)
  {
    cap=n;
    top=-1;
    m=new int[cap];
    st=new int[cap];
  }
  voidinput_marks()throws IOException
  {
    int i;
   BufferedReader br=new Buffered
   Reader(new Input Stream Reader
   (System.in));
   for(i=0;i<m.length;i++)
    {
      System.out.println("Enter a no.");
      m[i]=Integer.parseInt(br.readLine
      ());
       pushmarks(m[i]);
    }
  }

  void pushmarks(int v)
  {
    if(top==st.length-1)
    {
      System.out.println("Not
       Possible");
    }
    else
    {
      st[++top]=v;
    }
  }
  int popmarks()
  {
    if(top = = -1)
    {
      return(-999);
    }
    else
    {
      return(st[top--]);
    }
  }
  void display()
  {
    int i,x;
    for(i=top;i>=0;i--)
    {
      System.out.println(st[i]);
    }
  }
}
```

Question 13.

(a) *A linked list is formed from the objects of the class :*

```
class Node
  {
      int    number;
      Node   nextNode;
  }
```

Write an ***algorithm OR a Method*** *to add a node at the end of an existing linked list. The method declaration is as follows :*

void addnode (Node start, int num)

(b) *Define the terms* ***complexity*** *and* ***'O' notation.***

(c) *Answer the following from the diagram of the Binary Tree given below :*