**package** com.ssm.recurtation;

**import** java.util.Scanner;

**public** **class** SumOfNNaturalNumber

{

**public** **static** **void** main(String[] args)

{

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

System.***out***.println("enter the element");

**int** n=sc.nextInt();

sc.close();

**int** s=*sumOfN*(n);

System.***out***.println(s);

}

**static** **int** sumOfN(**int** n)

{

**if**(n<=1)

**return** 1;

**return** n+*sumOfN*(n-1);

}

}

**package** com.ssm.recurtation;

**import** java.util.Scanner;

**public** **class** SumOfDigit

{

**public** **static** **void** main(String[] args)

{

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

System.***out***.println("enter the element");

**int** d=sc.nextInt();

sc.close();

**int** s=*sumOfdigit*(d);

System.***out***.println(s);

}

**static** **int** sumOfdigit(**int** d)

{

**if**(d<1)

**return** 0;

**return** (d%10)+*sumOfdigit*(d/10);

}

}

**package** com.ssm.recurtation;

**import** java.util.Scanner;

**public** **class** SumOfArray

{

**public** **static** **void** main(String[] args)

{

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

System.***out***.println("enter the size element");

**int** size=sc.nextInt();

**int** ar[]=**new** **int**[size];

System.***out***.println("enter the "+size+"integer");

**for** (**int** i = 0; i < ar.length; i++)

{

ar[i]=sc.nextInt();

}

sc.close();

**int** s=*sumOfArr*(ar,0);

System.***out***.println(s);

}

**static** **int** sumOfArr(**int**[] arr, **int** i)

{

**if**(i==arr.length-1)

**return** arr[i];

**return** arr[i]+*sumOfArr*(arr, i+1);

}

}

**package** com.ssm.recurtation;

**import** java.util.Scanner;

**public** **class** RevereTheNumber

{

**public** **static** **void** main(String[] args)

{

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

System.***out***.println("enter the first element");

**int** m=sc.nextInt();

sc.close();

**int** r=*reversenu*(m,0);

System.***out***.println(r);

}

**static** **int** reversenu(**int** n, **int** rev)

{

**if**(n==0)

**return** rev;

**return** *reversenu*(n/10,rev\*10+n%10);

}

}

**package** com.ssm.recurtation;

**import** java.util.Scanner;

**public** **class** Prime

{

**public** **static** **void** main(String[] args)

{

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

System.***out***.println("enter the element");

**int** d=sc.nextInt();

sc.close();

**boolean** s=*primeOfNum*(d,2);

**if**(s)

System.***out***.println(d+"is prime");

**else**

System.***out***.println(d+ " not prime");

}

**static** **boolean** primeOfNum(**int** d,**int** i)

{

**if**(i>d/2)

**return** **true**;

**if**(d%2==0)

**return** **false**;

**return** *primeOfNum*(d,i+1);

}

}

**package** com.ssm.recurtation;

**import** java.util.Scanner;

**public** **class** Perfect

{

**public** **static** **void** main(String[] args)

{

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

System.***out***.println("enter the first element");

**int** m=sc.nextInt();

sc.close();

**boolean** n=*isPerfect*(m,1,0);

**if**(n)

System.***out***.println(m+" is perfect");

**else**

System.***out***.println(m+" is not perfect");

}

**static** **boolean** isPerfect(**int** n, **int** i, **int** sum)

{

**if**(i>n/2)

{

**return** sum==n;

}

**if**(n%i==0)

sum=sum+i;

**return** *isPerfect*(n, i+1, sum);

}

}

**package** com.ssm.recurtation;

**import** java.util.Scanner;

**public** **class** Penidrum

{

**public** **static** **void** main(String[] args)

{

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

System.***out***.println("enter the first element");

String st=sc.nextLine();

sc.close();

**boolean** rs= *isPelidrum*(st,0,st.length()-1);

**if**(rs)

System.***out***.println(st+" is pelindrum");

**else**

System.***out***.println(st+" not is pelindrum");

}

**static** **boolean** isPelidrum(String st, **int** i, **int** j)

{

**if**(i>=j)

**return** **true**;

**if**(st.charAt(i)!=st.charAt(j))

**return** **false**;

**return** *isPelidrum*(st,i+1,j-1);

}

}

**package** com.ssm.recurtation;

**import** java.util.Scanner;

**public** **class** NthPowerP

{

**public** **static** **void** main(String[] args)

{

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

System.***out***.println("enter the element");

**int** n=sc.nextInt();

**int** p=sc.nextInt();

sc.close();

**int** f=*power*(n,p);

System.***out***.println(f);

}

**static** **int** power(**int** n,**int** p)

{

**if**(p<=0)

{

**return** 1;

}

**else**

{

**return** n\**power*(n,p-1);

}

}

}

**package** com.ssm.recurtation;

**import** java.util.Scanner;

**public** **class** Lcm

{

**public** **static** **void** main(String[] args)

{

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

System.***out***.println("enter the first element");

**int** m=sc.nextInt();

System.***out***.println("enter the second element");

**int** n=sc.nextInt();

sc.close();

**int** l=*lcm*(m,n,m,n);

System.***out***.println(l+" is lcm");

}

**static** **int** lcm(**int** m, **int** n, **int** m1, **int** n1)

{

**if**(m1==n1)

**return** m1;

**if**(m1<n1)

**return** *lcm*(m,n,m1+m,n1);

**else**

**return** *lcm*(m,n,m1,n1+n);

}

}

**package** com.ssm.recurtation;

**import** java.util.Scanner;

**public** **class** Gcd

{

**public** **static** **void** main(String[] args)

{

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

System.***out***.println("enter the first element");

**int** m=sc.nextInt();

System.***out***.println("enter the second element");

**int** n=sc.nextInt();

sc.close();

**int** num= *gcd*(m,n);

System.***out***.println(num+" is gcd of "+ m+" and "+n);

}

**private** **static** **int** gcd(**int** m, **int** n)

{

**if**(m>n)

**return** *gcd*(n,m);

**if**(m==0)

**return** n;

**return** *gcd*(n%m,m);

}

}

**package** com.ssm.recurtation;

**import** java.util.Scanner;

**public** **class** Factorial

{

**public** **static** **void** main(String[] args)

{

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

System.***out***.println("enter the element");

**int** n=sc.nextInt();

sc.close();

**int** f=*factorial*(n);

System.***out***.println(f);

}

**static** **int** factorial(**int** n)

{

**if**(n<2)

{

**return** 1;

}

**else**

{

**return** n\**factorial*(n-1);

}

}

}