Date : 18/10/2022

Roll No. and Name : 22BCE538 Shah Kaivan

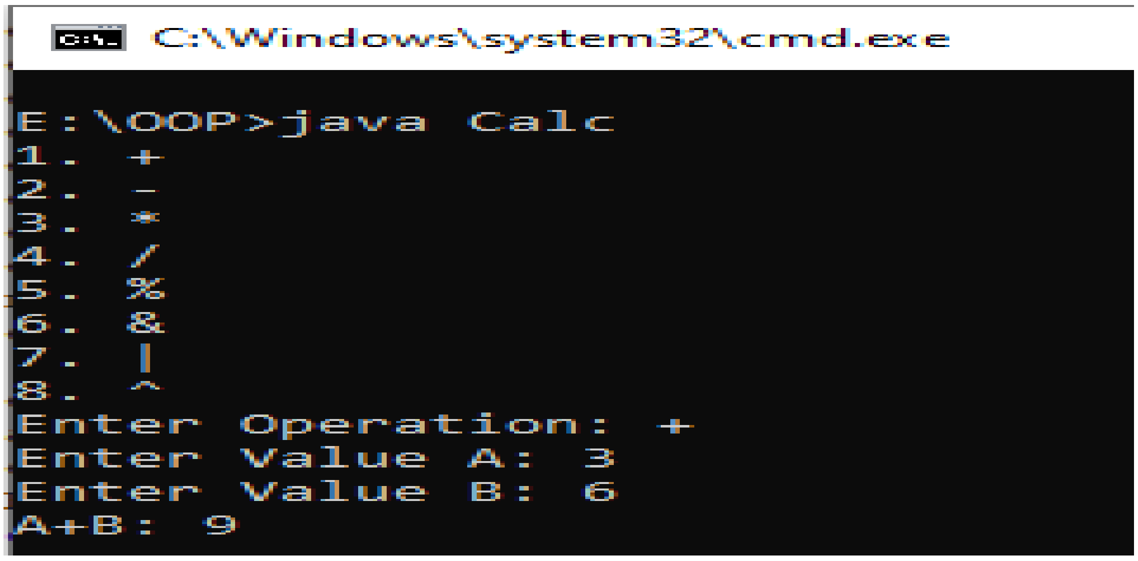
Course Code and Name : 2CS302 Object Oriented Programming

Practical No.: 3 (a)
AIM: Design calculator which contains arithmetic & bitwise operators. Operand(s) and operator must be scan from the user.
Methodology followed:

Input:

```
import java.util.*;

class Calc

{

    public static void main(String args[])

    {


        System.out.println("1. +");

        System.out.println("2. -");

        System.out.println("3. *");

        System.out.println("4. /");

        System.out.println("5. %");

        System.out.println("6. &");

        System.out.println("7. |");

        System.out.println("8. ^");

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter Operation: ");

        char o = sc.next().charAt(0);

        System.out.print("Enter Value A: ");

        int a=sc.nextInt();

        System.out.print("Enter Value B: ");
        int b=sc.nextInt();
        switch(o)
        {
            case '+': System.out.println("A+B: "+(a+b));
                    break;
            case '-': System.out.println("A-B: "+(a-b));
                    break;
            case '*': System.out.println("A*B: "+(a*b));
                    break;
            case '/': System.out.println("A/B: "+(a/b));
                    break;
            case '%': System.out.println("A%B: "+(a%b));
                    break;
            case '&': System.out.println("A&B: "+(a&b));
                    break;
            case '|': System.out.println("A|B: "+(a|b));
                    break;
            case '^': System.out.println("A^B: "+(a^b));
                    break;
        }
    }
}
```

Output:


C:\Windows\system32\cmd.exe

```
E:\OOP>java Calc
1.  +
2.  -
3.  *
4.  /
5.  %
6.  &
7.  |
8.  ^
Enter Operation: +
Enter Value A: 3
Enter Value B: 6
A+B: 9
```

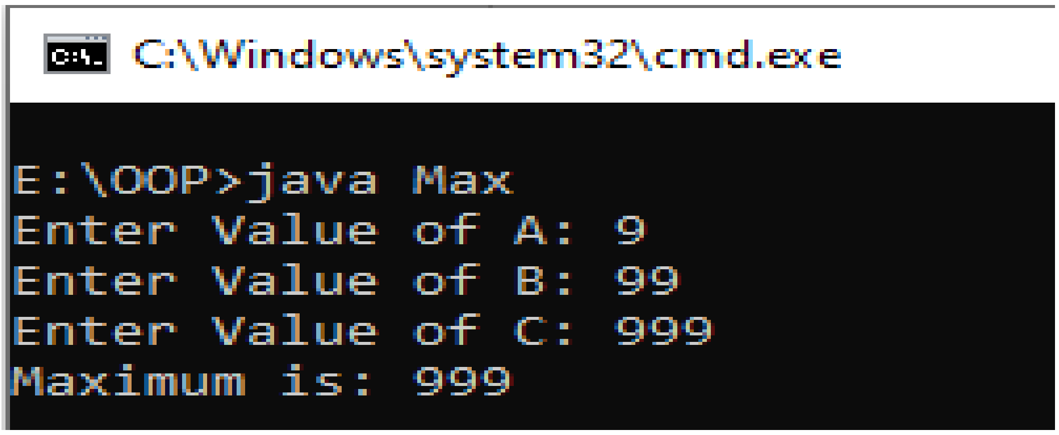
Conclusion :
I learnt that how to make a bitwise & arithmetic calculator using java.

Practical No.: 3 (b)
AIM: Find largest between three numbers using ternary operator.
Methodology followed:

Input:

```
import java.util.*;


class Max
{
    public static void main(String args[])
    {
        Scanner scan = new Scanner(System.in);
        System.out.print("Enter Value of A: ");
        int a = scan.nextInt();
        System.out.print("Enter Value of B: ");
        int b = scan.nextInt();
        System.out.print("Enter Value of C: ");
        int c = scan.nextInt();

        int max=(a>b)?((a>c)?a:c):((b>c)?b:c);

        System.out.println("Maximum is: "+max);
    }
}
```

Output:



```
E:\OOP>java Max
Enter Value of A: 9
Enter Value of B: 99
Enter Value of C: 999
Maximum is: 999
```

Conclusion :
I learnt to find the maximium element with the help of ternary operator.

Practical No.: 3 (c)
AIM: Given an array of size N-1 such that it only contains distinct integers in the range of 1 to N. Find the missing element.
Methodology followed:

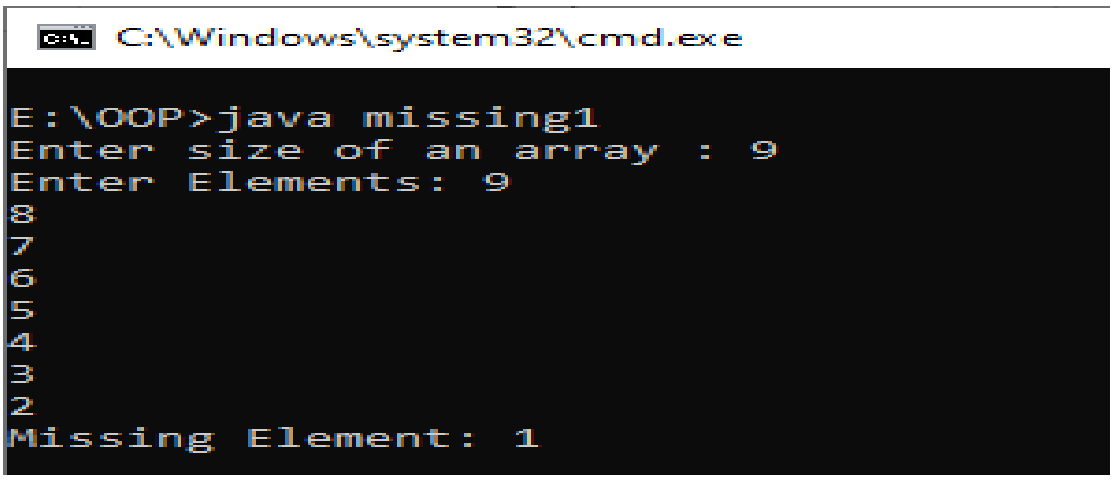
Input:

```
import java.util.*;
class missing1
{

        public static int missing(int[] arr)
        {
                int n=arr.length;
                int sum1=((n)*(n+1))/2;
                int sum2 = 0;
                for(int i=0;i<n;i++)
                {
                        sum2+=arr[i];
                }
                return sum1-sum2;
        }
        public static void main(String args[])
        {
                Scanner sc=new Scanner(System.in);
                System.out.print("Enter size of an array : ");
                int size=sc.nextInt();
                boolean isvalid = true;
                int a[]=new int[size];
                System.out.print("Enter Elements: ");
```

```
for(int i=0;i<size-1;i++)
{
        a[i]=sc.nextInt();
        if(a[i]<0 || a[i]>size)
        {
                isvalid=false;
                System.out.println("NEGATIVE OR GREATER
        THAN GIVEN NUMBER FOUND");
                break;
        }
        if(i>0)
        {
                if(a[i] == a[i-1])
                {
                        isvalid=false;
                        System.out.println("DUPLICATE Value");
                        break;
                }
                for(int j=0;j<i-1;j++)
                {
                        if(a[j]==a[i])
                        {
                                isvalid=false;
                             System.out.println("DUPLICATE Value ");
                                break;
                        }
                }
                if(isvalid==false)
```

```
                        {
                                break;
                        }
                }
        }

        if(isvalid)
        {
                System.out.println("Missing Element: " + missing(a));
        }
        sc.close();
    }
}
```

Output:



C:\Windows\system32\cmd.exe

```
E:\OOP>java missing1
Enter size of an array : 9
Enter Elements: 9
8
7
6
5
4
3
2
Missing Element: 1
```

Conclusion:

I learnt about how find missing element from 1 to n in O(1) time.

Practical No.: 3 (d)

AIM: Given an array of positive and negative numbers. Find if there is a subarray with 0 sum.

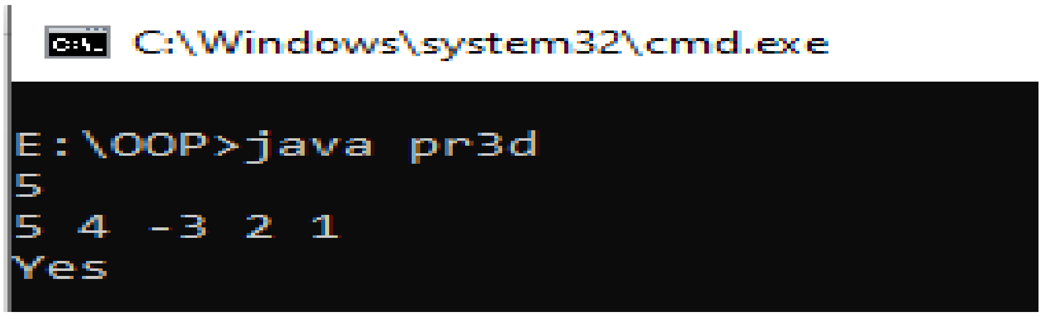
Methodology followed:

Input:

```
import java.util.Scanner;


class pr3d
{
  public static void main (String args[])
  {
    Scanner sc = new Scanner (System.in);
    int i, j, sum = 0, n = sc.nextInt ();
    int a[] = new int[n], mx = ((n * (n + 1)) / 2);
    for (i = 0; i < n; i++)
      {
          a[i] = sc.nextInt ();
      }
    int ans = 0;
    boolean check = false;


for (i = 0; i < n; i++)
      {
          ans = a[i];
          for (j = i + 1; j < n; j++)
            {
              ans = ans + a[j];
              if (ans == 0)
```

```
                {
                    check = true;
                    break;
                }
            }
          if (check)
            break;
       }

if (check)
       System.out.println ("Yes");
    else
       System.out.println ("No");
  }
}
```

Output:



C:\Windows\system32\cmd.exe

```
E:\OOP>java pr3d
5
5 4 -3 2 1
Yes
```

Conclusion :
I learnt how to check whether the sum is 0 in any subarray.

Practical No. : 3(e)

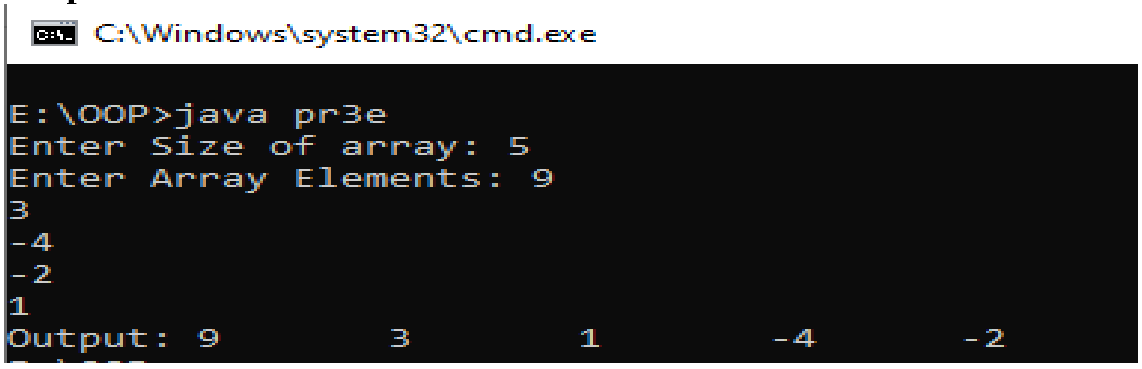
AIM: Given an unsorted array arr[] of size N having both negative and positive integers. The task is place all negative element at the end of array without changing the order of positive element and negative element.

Methodology followed:

Input:

```
import java.util.*;

 class pr3e
{
        public static void main(String args[])
        {
                int n,i;
                Scanner scan = new Scanner(System.in);
                System.out.print("Enter Size of array: ");
                n=scan.nextInt();
                int a[] = new int[n];
                System.out.print("Enter Array Elements: ");
                for(i=0;i<n;i++)
                {
                        a[i] = scan.nextInt();
                }
                System.out.print("Output: ");
                for(i=0;i<n;i++)
                {
                        if(a[i]>=0)
                        {
                                System.out.print(a[i]+"\t");
                        }
                }
                for(i=0;i<n;i++)
                {
                        if(a[i]<0)
                        {
                                System.out.print(a[i]+"\t");
                        }
                }
        }
}
```

Output:



```
E:\OOP>java pr3e
Enter Size of array: 5
Enter Array Elements: 9
3
-4
-2
1
Output: 9       3       1       -4      -2
```

Conclusion:
I learnt that how to print positive integer at first and negative integer at last.