IC 152 LAB ASSIGNMENT

Students-

Aayush Sankhe	B24281
Prateek Sharma	B24042
Rahul Garg	B24273
Vedant Naik	B24343
Amit Kumar	B24329
Amey Ballal	B24328

Group Learnings-

- Learning the use of bitwise operators (& , | , ^ , ~ , << , >>)
- Learning how to use print statement, and doing simple calculations on python.
- Learning how a decimal number is stored in 64 bit binary form.

1. (a) **Code-**

```
>>>print(2|3)
   3
>>>bin(2)
   '0b10'
>>>bin(3)
   '0b11'
```

Explanation - Bitwise and(|) of 2 and 3 is ‘0b11’ which is 3.

(b) **Code-**

```
n=int(input("Enter a natural number"))
if (n&1)==0:
      print("Number is even")
else:
      print("Number is odd")
```

Output-

```
>>>
= RESTART: C:/Users/prate/AppData/Local/Programs/Python/Python312/python/odd even.py
Enter a natural number42
Number is even
>>>
= RESTART: C:/Users/prate/AppData/Local/Programs/Python/Python312/python/odd even.py
Enter a natural number77
Number is odd
>>>
```

(c) **Code-**

```
>>>print(24>>1)
   12
```

2. (a) **Code-**

```
>>>print(27/8)
   3.375
```

(b) **Code-**

```
>>>print(27//8)
   3
```

(c) **Code-**

```
>>>print(27>>3)
   3
```

Explanation - Bitwise Right Shift operator (>>) shifts the bits

of binary notation of 27 (11011) to 3 places to the right, giving 00011 as result, which is 3 in decimal number system. (27>>3) effectively divides 27 by (2^3) which is 8, and rounds off to nearest integer. 27//8 also divides 27 by 8, and removes the remainder. Hence, the result is the same.

(d) **Code-**

```
>>>print(1<<10)
   1024
>>>print(1<<20)
   1048576
>>>print(2**10)
   1024
>>>print(2**20)
   1048576
```

(e) **Code-**

```
n=int(input("How many bits do you want in your binary
number"))
print("The binary number is", "1"*n)
b=0
for i in range(0,n):
     b=b+1*(2**i)
print("The decimal form of the number is", b)
print(bin(b))
```

Output-

```
>>>
= RESTART: C:/Users/prate/AppData/Local/Programs/Python/Python312/python/bin.py
How many bits do you want in your binary number4
The binary number is 1111
The decimal form of the number is 15
0b1111
>>>
= RESTART: C:/Users/prate/AppData/Local/Programs/Python/Python312/python/bin.py
How many bits do you want in your binary number6
The binary number is 111111
The decimal form of the number is 63
0b111111
>>>
```

3. (a) **Code-**

```
>>>print(-5,7)
-5 7
```

(b) **Code-**

```
>>>import math
>>>print(math.pow(1,3)+math.pow(12,3))
   1729.0
>>>print(math.pow(9,3)+math.pow(10,3))
   1729.0
```

4. (a) **Code-**

```
>>>print(50-3*10,(50-3)*10)
   20 470
```

(b) **Code-**

```
>>>import math
>>>math.log2(4096)
   12.0
```

(c) **Code-**

```
>>>a=2
>>>b=-7
>>>c=6
>>>D=(b**2)-4*a*c   #Discriminant
>>>root1=((-b)-D)/(2*a)
>>>root2=((-b)+D)/(2*a)
>>>print("The roots are",root1,",",root2)
   The roots are 1.5 , 2.0
```

5.

- The number is -14.125 , so the first digit will be 1, as the number is negative.
- Now, print(14.125/(2**3)) gives 1.765625 , so (e-1023)=3. Therefore e=1026. So 10000000010 is the 11 bit binary number.
- Now, (1+f)=1.765625, so f=0.765625 .
- print(0.765625-0.5) gives 0.265625, which is >=0. So 1^{st} bit of remaining 52 bits is 1.
- print(0.265625-0.25) gives 0.015625, which is >=0. So 2^{nd} bit of remaining 52 bits is 1.
- Now, 0.015625 is 2^{-6} , so the 3^{rd}, 4^{th} and 5^{th} bits of remaining 52 bits will be 0, and the 6^{th} will be 1. Remaining all other bits will be 0.
- So the 64 bit binary number for -14.125 will be 1100000000101100010000000000000000000000000000000000000000000000

Verification-

- Last 52 bits are 1100010000000000000000000000000000000000000000000000, so $(1*2^{-1} + 1*2^{-2} + 1*2^{-6})$ gives 0.765625, which is f. So (1+f)=1.765625
- Now, 11 bits after the first one are 10000000010. So, $e=(1*2^{10})+(1*2^{1})=1026$.
- $2^{e-1023} = 2^{1026-1023} = 2^{3} = 8$
- First number is 1, so s=1.
- Now, $(-1)^{1} * 8 * 1.765625 = -14.125$