Problems based on Recursion - 4

Assignment Solutions

PW SKILLS

Q1 - Given a number n. Print if it is an armstrong number or not. (Easy)
An armstrong number is a number if the sum of every digit in that number raised to the power of total digits in that number is equal to the number.

Input : 153

Expected Output : Yes

Explanation:

- First, we calculated a power function which calculates the power of any number a raised to the power b.
- If b = 0, it means that the power to which the number needs to be raised is zero, then the value is equal to 1 and this would be our base case from where we need to terminate this code.
- If the power is even then the number can be splitted in half power.
- If the power is odd then the number can be splitted in half power and 1 additional 'a' value can be multiplied externally.
- Create another function where we pass two parameters n(the original number) and dig(number of digits in the number) . Now we need every single digit of the number raised to 'dig' power. To extract any digit we can do n%10 this will give us the last digit of the number. But since we only want sum and nature of sum is commutative, that is in any order we do the sum it is independent of the order.
- After extracting the last digit we do not need it any more so we will pass the value n/10 to the next recursive call and this will be repeated until n does not turn out to be 0. Once n = 0 that means we have touched the base case where it shows that no digit is left in the original number.
- In the main function we calculate the number of digits present in 'n'. The number of times we are required to divide n by 10 till it becomes zero indicates the number of digits in that number. Eg. let n = 1234

Code:

https://pastebin.com/7ZHAkuBw

```
/Library/Java/JavaVirtualMachines/jdk-19.j
Enter the number n:
153
yes

Process finished with exit code 0
```

Q2 - Given two number x and y find product using recursion. (Easy)

Input : x = 5, y = 2
Expected Output : 10

Explanation:

- If x is less than y, swap the two variables value
- Recursively find y times the sum of x
- If any of them become zero, return 0

Code:
https://pastebin.com/FbpUxX8P

```
/Library/Java/JavaVirtualMachines/jdk-19.j
Enter the numbers:
5 2
10

Process finished with exit code 0
```

Q3 - Given a number n, check whether it's a prime number or not using recursion. (Easy)

Input : n = 11
Output : Yes

Explanation:

- We use the general algorithm to check if a number is prime.
- A number is prime if it cannot be divided entirely(with remainder 0) by any number other than 1 and itself.
- This can be shortened by dividing the number from 1 till i where i*i <= n.
- We call a recursive function with n and i as parameters, i representing the divisor and initialized with 2.
- If at any point, n%i becomes 0, we return false as the number is not prime.
- We stop when i*i exceeds n.

Code:
https://pastebin.com/2tEmt683

```
/Library/Java/JavaVirtualMachines/jdk-19.j
Enter the number n:
11
Yes

Process finished with exit code 0
```

Q4 - Given a decimal number as input, we need to write a program to convert the given decimal number into its equivalent binary number. (Easy)

Input : 7

Expected Output : 111

Explanation:

- 2 | 7 1 ↑
  2 | 3 1 |
  2 | 1 1 |
- Arrange the remainders in reverse fashion, so binary number becomes 111.
- Recursively we perform the first step, and call recursion for all others.

Code:

https://pastebin.com/gfrX3L0w

```
/Library/Java/JavaVirtualMachines/jdk-19.j
Enter the number n:
7
111

Process finished with exit code 0
```

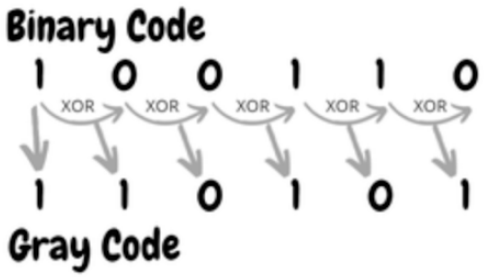
Q5 - Given the Binary code of a number as a decimal number, we need to convert this into its equivalent Gray Code. In gray code, only one bit is changed in 2 consecutive numbers. (Medium)

Input: 1001

Expected Output: 1101

Explanation:

- Binary to Gray conversion :
  The Most Significant Bit (MSB) (the first digit) of the gray code is always equal to the MSB of the given binary code.
  Other bits of the output gray code can be obtained by XORing binary code bit at that index and previous index.



- In recursive function, we check whether the last bit and second last bit are same or not, if it is same then move ahead otherwise add 1 as for XOR, if two bits are different, only then the output is true or 1.

Code:
https://pastebin.com/k6cHwuEv

```
/Library/Java/JavaVirtualMachines/j
Enter the binary number n:
1001
1101

Process finished with exit code 0
```