Jake Tantorski CMPT435 February 4th,2020

1.

a. My idea behind this algorithm is that while the number that you are trying to reverse is not zero, which would mean it cannot be reversed and also gives a way for the program to run until the last digits have been calculated. In the first line of the while loop, I took the reverse number and multiped it by 10 in order to move each number up a place value. The reason for this is the next number I will add to that reverse number is the next number in the sequence. I am able to get this number by taking the original number and modulo ten from it. This will give me the final digit of the current number being reversed. From there I divide the original number by 10 to get to the next place value. Overall, I am taking the last digit of the current number being evaluated and moving it to the beginning of a new variable called reverse number that can store the number reversed.

b.
```
Input int num
reverseNumber = 0
while (num != 0)
        reverseNumber = reverseNumber * 10;
        reverseNumber = reverseNumber + num%10;
        num = num/10;
end while
Output reverseNumber
```

2.

a. My idea for this design was in order to find the missing number I would be able to add up all of the numbers within the given array and take that sum and subtract it from my total formula for n. This formula allows the function to calculate the total sum for the length of 1through n. By knowing what the total of the array should be if the missing number was in there, you can subtract the total for the current array thus the difference would the missing number.

b. **Input** A[]

```
sum = 0, total = (A.length+2)*(A.length+1)/2
for i in [0,size(A)]
    sum = sum + A[i]
end for
missng = total - sum
Output missing
```

3.

a. My idea for this design was in order for there to be a majority the first integer would have to be the same as the middle depending on the location of the index. I used a for loop in order to compare the values at each index to determine if there was a majority present or not.

b. **Input** A[]

```
n = A.length
half = n/2
for i in [0,half]
    if(A[i] == A[i+n/2])
        Output true
    end if
end for
Output false
```