In [ ]:
```
# Problem 1
```

In [57]:
```
# map input DNA strand to a complementary sequence in the same order
def CompleDNA(dnaSeq):
    """
    This function takes a string of DNA base pairs and
    returns a complement of a DNA strand.
    """
    pairs = {'G':'C','C':'G','A':'T','T':'A'} #DNA base pairs
    c_dna = ''  #complementary DNA sequence

    for letter in dnaSeq:
        if letter not in pairs:
            print('Not a DNA sequence. Please try again!')
            break
        else:
            c_dna += pairs[letter]
    print(c_dna)
```

In [58]:
```
CompleDNA('BA')
```

```
Not a DNA sequence. Please try again!
```

In [63]:
```
# Update the function to take both upper and lower case letters
def CompleDNA(dnaSeq):
    """
    This function takes a string of DNA base pairs and
    returns a complement of a DNA strand.
    """
    pairs = {'G':'C','C':'G','A':'T','T':'A'} #DNA base pairs
    c_dna = ''  #complementary DNA sequence
    dnaSeq = dnaSeq.upper() # Converting sequences to uppercase

    for letter in dnaSeq:
        if letter not in pairs:
            print('Not a DNA sequence. Please try again!')
            break
        else:
            c_dna += pairs[letter]
    print(c_dna)
```

In [64]:
```
CompleDNA('aatc')
```

```
TTAG
```

In [65]:
```
# Update the function with a default argument direction='same'
# reverse the sequence if user sets the direction to 'reverse'

def CompleDNA(dnaSeq, direction='same'):
    """  This function takes a string of DNA base pairs and
    returns a complement of a DNA strand.
    Set the direction to 'reverse' if you want to get a reversed strand.
    """
    pairs = {'G':'C','C':'G','A':'T','T':'A'} #DNA base pairs
    c_dna = ''  #complementary DNA sequence
    dnaSeq = dnaSeq.upper() # Converting sequences to uppercase
    for letter in dnaSeq:
        if letter not in pairs:
            print('Not a DNA sequence. Please try again!')
            break
        else:
            c_dna += pairs[letter]

    # reverse the sequence if user sets the direction to 'reverse'
    if direction == 'reverse':
        print('Reversed complementary DNA sequene:',c_dna[::-1])     # or ''.jo
    else:
        print(c_dna)
```

In [66]:
```
CompleDNA('gattaca')
```

```
CTAATGT
```

In [67]:
```
CompleDNA('gattaca','reverse')
```

```
Reversed complementary DNA sequene: TGTAATC
```

In [72]:
```
# Problem 2
# Write a program that asks user to enter an integer, and then
# reports whether it is a prime number or not.
def prime():
    print('This program determines if a number is a prime number.')
    num = eval(input("Enter a number: "))

    if num>1:
        for i in range(2,num):
            if num % i == 0:
                print(num,"is not a prime number.")
                break
        else:
            print(num,"is a prime number.")
    else:
        print(num,"is not a prime number. Your number have to be a positive int
```

In [73]:
```
prime()
```

```
This program determines if a number is a prime number.

0.1 is not a prime number. Your number have to be a positive integer.
```

In [74]:
```
prime()
```

```
This program determines if a number is a prime number.

27 is not a prime number.
```

In [ ]:
```

```