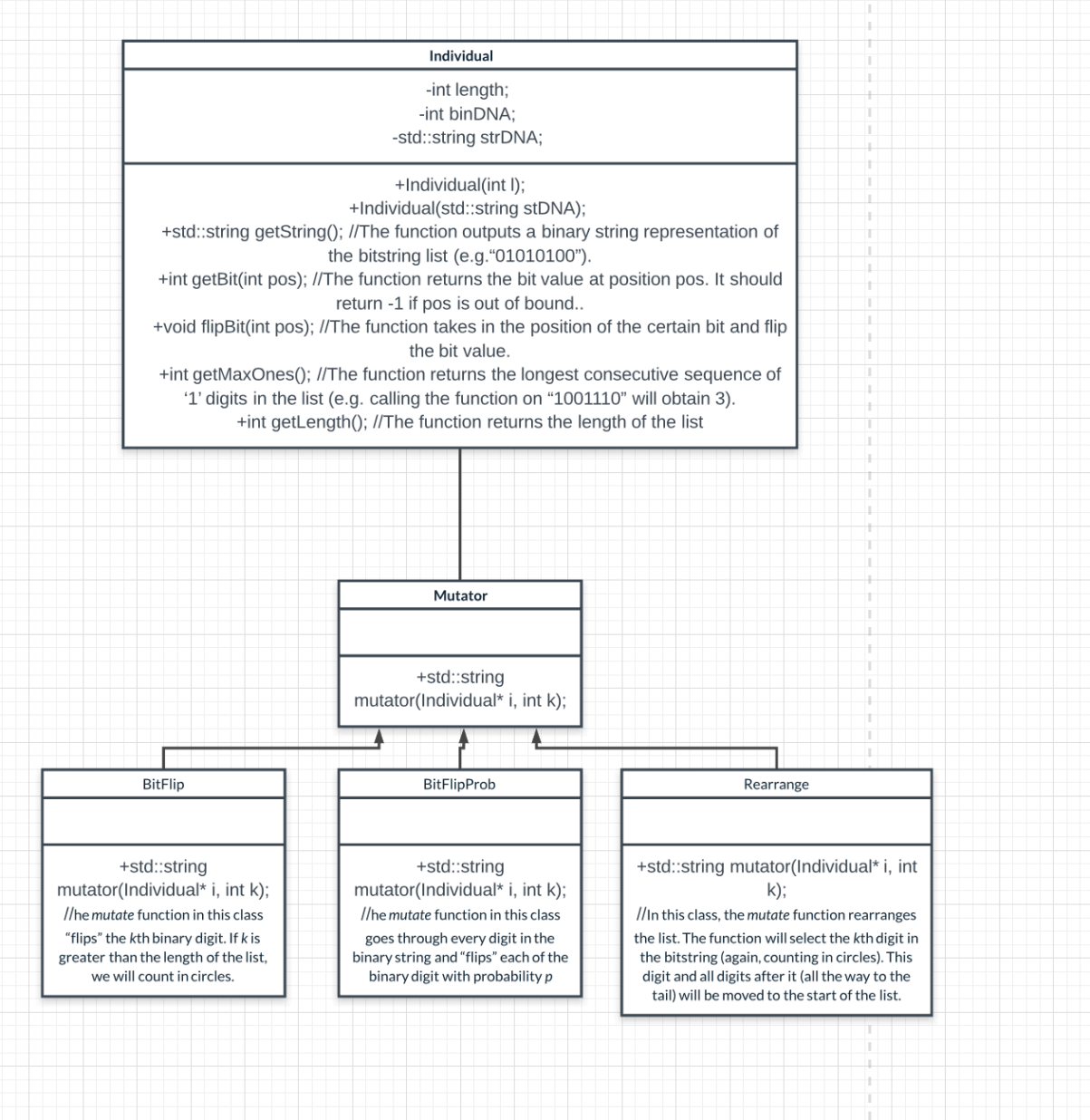
Design

Yian Xie
A1702241



Description:

Individual:

- string getString() : The function outputs a binary string representation of the bitstring list (e.g.“01010100”).
- int getBit(int pos) : The function returns the bit value at position pos. It should return -1 if pos is out of bound..
- void flipBit(int pos) : The function takes in the position of the certain bit and flip the bit value.
- int getMaxOnes() : The function returns the longest consecutive sequence of ‘1’ digits in the list (e.g. calling the function on “1001110” will obtain 3).
- int getLength() : The function returns the length of the list.
- A constructor that takes in the length of the binary DNA and creates the the binary string. Each binary value in the list should be given a value of 0 by default.
- A constructor that takes in a binary string and creates a new Individual with an identical list. Note that this involves creating a new copy of the list.

Mutator:

+std::string mutator(Individual* i, int k);

BitFlip: this class inherits individual class

+std::string mutator(Individual* i, int k);//he *mutate* function in this class “flips” the *k*th binary digit. If *k* is greater than the length of the list, we will count in circles.

BitFlipProb: this class inherits individual class

+std::string mutator(Individual* i, int k);//he *mutate* function in this class goes through every digit in the binary string and “flips” each of the binary digit with probability *p*

Rearrange: this class inherits individual class

+std::string mutator(Individual* i, int k);//In this class, the *mutate* function rearranges the list. The function will select the *k*th digit in the bitstring (again, counting in circles). This digit and all digits after it (all the way to the tail) will be moved to the start of the list.

Testing:

Input: `111000111 1 0000111110000 5`
Expected output: `011000111 1111100000000 5`
Output: `011000111 1111100000000 5`

Input: `11 3 01 1`
Expected output: `01 01 1`
Output: `01 01 1`

Input: `1101 3 10001 5`
Expected output: `1111 01100 2`
Output: `1111 01100 2`

Input: `00000 15 00110 2`
Expected output: `00001 01100 2`
Output: `00001 01100 2`