The design of assignment 6

Binaryclass
+ data: bool + head Binaryclass* + next Binaryclass*
+ insrt(bool number) void

BinaryNode
+ one_zro bool + head_node Binaryclass* + next_node Binaryclass*
+ insrt_Node(bool number) void

Individual: + BinaryNode
+ getString() string + getFirstBit() BinaryNode* + setFirstBit(BinaryNode* newHead) void + getMaxOnes() int + getLength() int + Insert(bool num) void + virtual void execute(int k) + ~Individual() + Individual()

BitFlip: + Individual
void execute(int k)

Rearrange: + Individual
void execute(int k)

Design

1. Create a Individual class which inherit from the BinaryNode class, and we can use the variable of BinaryNode in Individual class.
2. getString() is for out put the value of each node, so we only need to great a node and go through the string and print out all the element of each node.
3. BinaryNode* getFirstBit() is for getting the 1st node of the string, and we only have one head node so just return head_Node.
4. void setFirstBit(BinaryNode* newHead) is for set the 1st node, so we need a template node point to the newHead and return.

5. getMaxOnes() is for printing longest consecutive sequence of ’1’ and we need to set a node and go thought all the string, we need another 2 variable, sum and max which means the template largest number and largest number. We need a for loop, if 1 is consecutive, sum=sum+1; until 0 exist. And we will compare sum and max, get the largest number set in to the max variable, then return max.
6. getLength() is for getting the length of the string, we just add a variable length and go through all the node ,every time, length= length+1; and return.
7. BitFlip: we do not need to think about the circles problem because in the main function we can use for loop to solve this problem. So in this function we need to set a template node and go to the kth position, and change the value from 0 to 1 or 1 to 0. And delete the original kth node.
8. Rearrange: we need to find the kth position and delete the node of k-1th element, then set the head node connected by the last node.

Testing and Case:

The length of string, 1st node and last node and the middle nodes are the point we need to test.

What we need to test is in below:

- If K<length of the string, change the middle element from 0 to 1;
  Input: 000000 1 0000 1
  Expect: 100000 0000
- If K<length of the string, change the 1th element from 0 to 1;
  Input: 000000 3 0000 1
  Expect: 001000 0000
- If K<length of the string, change the last element from 0 to 1;
  Input: 000000 6 0000 1
  Expect: 000001 0000
- If K<length of the string, change the kth element from 1 to 0;
  Input: 001000 3 0000 1
  Expect: 000000 0000
- If K>length of the string, change the kth element from 0 to 1;
  Input: 000000 17 0000 1
  Expect: 000001 0000

In this function we only need to test the length of string because we delete the node, whatever the 1st or the last, they get the same principle.

What we need to test is in below:

- If K<length of string change the kth element after string;
  Input: 000000 1 1011 3
  Expect: 000000 1110
- If K>length of string change the kth element after string

Input: 000000 1 1011 14
Expect: 000000 0111