Ethan Henry

Asgn4 DESIGN.pdf

PreLab:

1. Same as below, but convert to base 10 first using

```
get_base(int i)
    for(int i = 0; i > 0; i++)
        Num = i % 10
        I = i / 10;
    Return num;
```

2. Mercennes:

For every prime

    Plug in prime to 2^n - 1;
    Set that bit to 1;

Lucas:

For every prime

    Set up array, array[0] = 2, array[1] = 1;
    Proceed with normal calculation of Lucas numbers
        Test if every lucas number is equal to prime number.

Fibonacci:

For every prime

    Set up array, array[0] = 0, array[1] = 1;
    Calculate fibonacci numbers
        Test if each fibonacci number is prime.

1. BitVector Pseudocode

```
bv_create(uint32_t size)
    BitVector *v = malloc(size);
    v->length = size;
    v->vector = malloc(size);
    Return v;

bv_delete(BitVector *v)
    free(v);

bv_get_len(BitVector *v)
    Return v->length;
```

```
bv_clr_bit(BitVector *v, uint32_t i)
        Flag = 1;
        Flag = ~flag;
        Pos = i / 8
        Num = i % 8;
        Flag = flag << pos;
        v->vector[num] = v->vector[num] & flag;

bv_set_bit(BitVector *v, uint32_t i)
        Pos = i / 8
        Num = i % 8;
        Flag = 1;
        Flag = flag << pos;
        v->vector[num] = v->vector[num] | flag;

bv_set_all_bits(BitVector *v)
        Flag = 0;
        Flag = ~flag;
        For (int i  = 0; i < size; i ++)
                v->vector[i] = v->vector[i] | flag;
```

2. Test if the memory allocated is null. Also, make sure to free the ADT afterwards.
3. You could consolidate some of the code that is repeated into 1 line of code and call it with a variable. This will mean that the compiler will only have to calculate it once and then use the value over and over again.

PSEUDOCODE:

```
OPTIONS “nps:
Switch (c)
        Case ‘n’:
                BitVector *bit = bv_create(size);
                BitVector *mer = bv_create(size);
                BitVector *luc = bv_create(size);
                BitVector *fib = bv_create(size);
        Case ‘s’:
                For (int i = 0; i < bv_get_len(bit); i++)
                        If (bv_get_bit(bit, i) == 1)
                                Is_palindrome(i);
        Case ‘p’:
```

```
For (int i = 0; i < bv_get_len(bit); i++)
        If (bv_get_bit(bit, i) == 1)
                printf("prime");
                If (bv_get_bit(mer, i))
                        printf("mercennes");
                If (bv_get_bit(luc, i) == 1)
                        printf("lucas");
                If (bv_get_bit(fib, i) == 1)
                        printf("fibonacci");
```

```
Bool is_palendrome(int32_t i)
        Left = 0;
        Right = bv_get_len(bit);
        For (left < right)
                If (left != right)
                        Return false;
        Return true;
```