Artyom Martirosyan
10/29/2019
Assignment 5: factoring
Design document

V.0

For this assignment we will be using several algorithms to determine whether or not a number is prime or composite, if the number is prime the program will simply print out P, but if the number is a composite the the program will print out the prime numbers used to create the number(if numbers multiplied you will get the composite number). The program will be tested to a default of 100,000 unless a user argument is provided. To start off I will have a while loop with checking for the -n argument and if the argument is provided I will change the default number to selected. From there I will divide the number by 2 and make a bit vector of that number, after making the bit vector I will send the bit vector value through the sieve operator given to us by the professor. After the sieve operation I will create another loop from 1 to half of value selected which will check if the number in the bit vector is prime if so I will have an if statement which will be value selected mod prime number, if the remainder is zero then I will divide by that number, print the prime number, then subtract selected number by 1(just incase the number can be divided by that prime number again such as 4) and it will continue to check until the number is 1 meaning that is has been factored in as much as it can and so on and so forth.

V2.0(hoping the mulligan is accepted)

BV.C code:

Structs used:

    Bv struct holding bv->length and bv->vector

```
BitVector *bv_create(uint32_t bit_len) {
        Bitvector *v = malloc…(size of bv)
        If size %8 = 0 {
                Do a calloc for v->vector using length/8 as length
        { else {
                Do a calloc for v->vector using length/8 +1 as length
        }
        Set vector length variable to length
}
void bv_delete(BitVector *v) {
        Free v->vector
        Free v
}
uint32_t bv_get_len(BitVector *v) {
        Return length
}
void bv_set_bit(BitVector *v, uint32_t i) {
  v->vector[i / 8] |= (0x1 << (i % 8));
  return;
}
```

```
void bv_clr_bit(BitVector *v, uint32_t i) {
 Same logic as set bit but do a &= operation as wel as ~infront of the shifted value 0x1
}
uint8_t bv_get_bit(BitVector *v, uint32_t i) {
  uint8_t bit = (v->vector[i / 8] & (0x1 << (i % 8))) >> (i % 8);
  return bit;
}

void bv_set_all_bits(BitVector *v) {
  Do same logic for creating the vector using calloc(if mod 8 = 0 do this else...)
  mset(v->vector, 0xff, byte_length);
}
```

Sieve.c

Note the program was provided by darrell long therefore I am sourcing him here and providing the code below:

```
void sieve(BitVector *v) {
  bv_set_all_bits(v);
  bv_clr_bit(v, 0);
  bv_clr_bit(v, 1);
  bv_set_bit(v, 2);
  for (uint32_t i = 2; i <= sqrtl(bv_get_len(v)); i += 1) {
    if (bv_get_bit(v, i)) {
      for (uint32_t k = 0; (k + i) * i <= bv_get_len(v); k += 1) {
        bv_clr_bit(v, (k + i) * i);
      }
    }
  }
  return;
}
```

Factor.c

```
Main(getopt vars) {
Run getopt checking the different options(size)
Note if no size given have a default of 100000
Create the bitvector(using size)
Run the sieve
for(i=2 going to size) {
        if(getbit(i)) {
                Int temp = i
                for(z=2 - i) {
                        if(getbit(z)) {
                                if(temp %z == 0) {
```

```
                printf(z)
                Temp = temp / z;
                }
            }
        }
    } else {
     Print C
    }
}
}
```