Din Bostandzic
CSE 13S

Assignment 4 - Bit Vectors and Primes Program Design

Pre-Lab Answers

1. Pseudocode to determine if number is Fibonacci/Lucas/Mersenne prime

```
Begin is_fib module
    Begin For
    For (integer i initialized to 0, i < total, increment i)
        Declare integer a, assigning value of fib(in num as integer)
        Begin if
        If (num == a)
            Return true
        End if
    End for
End is_fib

Begin is_lucas module
    Begin For
    For (integer i initialized to 0, i < total, increment i)
        Declare integer a, assigning value of lucas(in num as integer)
        Begin if
        If (num == a)
            Return true
        End if
    End for
End is_lucas

Being is_merse module
    Begin for
    For (integer i initialized to 2, i < = total, increment i)
        Declare integer a, assigning value of merse(in num as integer)
        Begin if
        If (num == a)
            Return true
        End if
    End for
End is_merse
```

2. Pseudocode to determine if number in any base is pseudocode

```
Begin is_Palindrome module
    Declare f as bool initialized to true
    Declare length as integer initialized to strlen(in s as string)
```

```
        Begin for
        For (integer i initialized to 0, i < length, i++)
            Begin if
            If (s[i] != s[length -(i+1)])
                Assign to f value of false
            End if
        End for
    Return f
    End is_Palindrome
```

Data design

```
Define OPTIONS as string constant "spn:"
Declare next_input as string initialized to NULL
Declare default_num as integer initialized to 0
Declare s, p, n as bool initialized to false
Declare c as integer initialized to 0
```

Main module design

```
Begin Main (pass in argc as integer, in argv as string)
    Begin While
    While (c = getopt(pass in argc as integer, in argv as string, in OPTIONS as
        string)) does not equal -1
        Begin switch (c)
            Case 's'
                Assign value of true to s
                Break statement
            Case 'p'
                Assign value of true to p
                Break statement
            Case 'n'
                Assign value of true to n
                Assign value of  optarg to next_input
                Assign value of next_input converted to integer, to default_num
                Break statement
            Default Case
                Display "Character not defined in the string"
                Return with exit status fail
        End switch
    End While
    Begin if
    If (argc == 1)
        Display “Error: no arguments supplied!”
        Return with exit status fail
    End If
    Begin if
    If(s == true)
        Begin If
        if(n == false)
            Assign to default_num value of 1000
        End if
        Call prime_all_num()
    End if
    Begin if
    If(r == true)
        Begin If
        if(n == false)
            Assign to default_num value of 1000
        End if
        Call palindrome_prime()
    End if
End Main
```

<u>bv.c implementations design</u>

<u>bv_create module design</u>

```
Begin bv_create(int size)
    Dynamically allocate v of type BitVector structure
    BitVector *v = (struct BitVector*)malloc(sizeof(BitVector));
    if(v == NULL)
    Return 0
    If (size < 1)
    Size = 1;
    v->length = size
    v->vector = (int*)malloc(sizeof(int) * size);
    if(v->vector == NULL)
    Return 0
Return v
End module
```

```
Begin void bv_delete ( BitVector *v)
        Call free(v->items)
        Call free(v)
        return
End module

Begin uint32_t bv_get_len ( BitVector *v);
        Return v->length
End module
```

Sets the bit at index in the BitVector .
/ // v : The BitVector .
// i : Index of the bit to set .

```
void bv_set_bit ( BitVector *v, uint32_t i)
        uint8_t bits = v->vector[i/32];
          printf("length of vector: %d\n", bits);
        uint8_t newbit = (00000001<<i);
        base_to_bin(newbit);
        printf("length of newbit: %d\n", newbit);
        uint8_t newresult = bits | newbit;
        printf("length of vector: %d\n",newresult);
End module
```

//i : Index of the bit to clear

```
 void bv_clr_bit ( BitVector *v, uint32_t i)
        uint8_t bits = v->vector[i/32];
          printf("length of vector: %d\n", bits);
        uint8_t newbit = ~(00000001<<i);
        base_to_bin(newbit);
        printf("length of newbit: %d\n", newbit);
        uint8_t newresult = bits & newbit;
        printf("length of vector: %d\n",newresult);
End module
```

```
uint8_t bv_get_bit ( BitVector *v, uint32_t i)
        Return uint8_t bit = v->vector[i%32];

 void bv_set_all_bits ( BitVector *v);
    for (int i  = 0; i < bv_get_len ( BitVector *v); i++)
                bv_set_bit ( v, i)
    End for
End module
```