Problem-Set

Submitted To:

Subroto Nag Pinku

Department of CSE

Daffodil International University

Submitted By:

Rubaiya Tasnim Sholi

ID : 191-15-12361

Section : O-14

Department of CSE

Daffodil International University

Part B:
Task 5: Solution:

```
#include <stdio.h>

unsigned long long fibo(int num);



int main()
{
    int num;
    unsigned long long fibonacci;
    scanf("%d", &num);

    fibonacci = fibo(num);

    printf("%d fibonacci term is %llu", num, fibonacci);

    return 0;
}

unsigned long long fibo(int num)
{
    if(num == 0)
        return 0;
    else if(num == 1)
        return 1;
    else
        return fibo(num-1) + fibo(num-2);
}
```

Analysis:

1st number ->0

2nd number->1

N th fibo = sum previous 2 number

3rd = 1st + 2nd = 0 + 1 = 1

4th = 3rd + 2nd = 1 + 1 = 2

5th = 4th + 3rd = 2 + 1 = 3

```
Int fibo[10];      //array size
fibo [0] = 0;     //array index
fibo [1] = 1;    //array index
fibo [2] = fibo [2 - 1] + fibo [2 - 2] = 1 + 0 = 1
fibo [3] = fibo[3 - 1] + fibo [3 - 2]
          = fibo[2] + fibo[1] = 1 + 1 = 1
fibo[4] = fibo[4 - 1] + fibo[4 - 2]
          = fibo[3] + fibo[2] = 2 + 1 = 3
```

```
Int fibo[11];
Fibo[0] = 0;
Fibo[1] =1;
for(i=2; i<11; i++)  //n=10
 Fibo[i] = fibo[i - 1] + fibo[i - 2]
 Fibo[3] = fibo[3 - 1] + fibo[3 - 2]
            = fibo[2] + fibo[1] = 1 + 1 = 2
```

nth fibo number time complexity O(N)

Task 6:Solution:

```
#include <stdio.h>
#include<math.h>
#include<string.h>
int main()
{
  int ara[1001],x;
  ara[0]=0;
  ara[1]=1;
  int n,rem;
   for(int i=2;i<1000;i++){
      ara[i]= (ara[i-1] +ara[i-2] )%10;
  }
  scanf("%d",&n);
  printf("%d",ara[n]);
}
```

Analysis:

Int fibo[11]

Fibo[0] = 0;

Fibo[1] = 1;

for(int i = 2; i<11; i++) //n time loop

Fibo[i] = (fibo [i - 1] + fibo[i - 2] % 10;

15 % 10 = 5 // % modulus

Time Complexity: O(n)

I = 2

Fibo[2] = (fibo[2 - 1] + fibo[2 - 2] % 10;

= (fibo[1] + fibo[0]) % 10

= (1 + 0) % 10 = 1 % 10 = 1

Fibo[2] = 1

Fibo[7] = 3

Fibo[8] = 1

Fibo[9] = 4

Task 7 : Solution:

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int i, num1, num2, min, hcf=1;

    scanf("%d%d", &num1, &num2);

    min = (num1<num2) ? num1 : num2;

    for(i=1; i<=min; i++){
        if(num1%i==0 && num2%i==0){
            hcf = i;
        }
    }

    printf("HCF of %d and %d = %d\n", num1, num2, hcf);

    return 0;
}
```

Analysis:

Here we can take two numbers a & b

gcd of a & b:

While b is not equal 0

p = b

b’ = a mod b

a’ = p

Here, it divides the big number by a small number. If the remainder is not equal to 0, it goes to next step. Then divide the divisor by the remainder. Repeat the process until the remainder is equal to 0, when the remainder is 0 that divisors are gcd.

Suppose we have 24 & 18 .At first, dividing the large number 24 by the small number 18. Here 24/18 = 1 & reminder 6. Then dividing the divisor 18 by the reminder 6, 18/6=3 reminder 0. So, gcd of 24 & 18 is 6.

Time complexity : int gcd(int a, int b){
If (a == 0) return b;
return gcd(b%a,a)
}

So the time complexity is log(min(a,b))

Task 8: Solution:

```
#include <stdio.h>
int main()
{
   int n1, n2, min;
   scanf("%d %d", &n1, &n2);

   min = (n1 > n2) ? n1 : n2;

   while (1) {
      if (min % n1 == 0 && min % n2 == 0)
      {
         printf("The LCM of %d and %d is %d.", n1, n2, min);
         break;
      }
      ++min;
   }
   return 0;
}
```

Analysis:

Time complexity : O(log(min(a,b))) + O(1)

= O(log(min(a,b)))

As an explanation we can say that , a relation between the gcd & lcm of two numbers is that the product of the two numbers is equal to the product of their gcd & lcm. As an example, consider 15 and 20: the gcd is 5, the lcm is 60, 15 × 20 = 300 and gcd × lcm = 300 or lcm(a,b) = a* b / gcd(a,b), So here the time complexity is O(log(min(a,b)))

hkkjx