RSA Public-Key Cryptosystem

1. Mahmoud Emad Abdellatif (15)
2. Mahmoud Elsayed Mohammed Nasr Eldeen (15)
3. Mahmoud Saeed Mohammed Hasan (15)
4. Rofayda Abdelmaksoud (7)

Team ID: - 43

1. Addition Function:-

```
        public string Addtion(string first, string second)      → O(N)
{
  if(second.Length>first.Length)   → O(N)
  {
     string tmp = first; → O(N)
     first = second;    → O(N)
     second = tmp;      → O(N)
  }
  int count;     → O(1)
  int carry=0;   → O(1)
  int len1 = first.Length;    → O(1)
  int len2 = second.Length;   → O(1)
  StringBuilder add_rusIt = new StringBuilder(first); → O(1)
  count = len1 - len2;  → O(1)
  for (int j = len2 - 1; j >= 0; j--)   → O(N)
  {
     add_rusIt[j + count] = (char)((first[j + count] - '0') + (second[j] - '0') + carry + 48);   → O(1)
     if ((add_rusIt[j+count]-'0') >= 10)     → O(1)
     {
        int mod = (add_rusIt[j + count] - '0') % 10;    → O(1)
        int div = (add_rusIt[j + count] - '0') / 10;    → O(1)
        add_rusIt[j + count] = (char)(mod + 48);        → O(1)
        carry = div;                                    → O(1)
     }
     else   → O(1)
     {
        carry = 0;     → O(1)
     }
  }
  for (int j = count - 1; j >= 0; j--)    → O(N)
  {
     add_rusIt[j] = (char)((first[j] - '0')+carry+48);   → O(1)
     if ((add_rusIt[j]-'0') >= 10)     → O(1)
     {
        int mod = (add_rusIt[j]-'0') % 10;  → O(1)
        int div = (add_rusIt[j]-'0') / 10;  → O(1)
        add_rusIt[j] = (char)(mod+48 );     → O(1)
        carry = div;                        → O(1)
     }
     else    → O(1)
     {
        carry = 0;  → O(1)
     }
  }
  if(carry>0)     → O(1)
  {
     return carry.ToString() + add_rusIt.ToString();     → O(1)
  }
  if(add_rusIt[0]!='0')   → O(N)
  {
     return add_rusIt.ToString();    → O(N)
  }
  count = 0;              → O(1)
  bool finsh = false;     → O(1)
  for (int i = 0; i < len1; i++)  → O(N)
  {
     if (add_rusIt[i] != '0')    → O(1)
     {
        finsh = true;   → O(1)
        break;          → O(1)
     }
     else    → O(1)
     {
        count++;    → O(1)
     }
  }
  if(finsh) → O(N)
```

```
    {
        return add_rusIt.ToString().Substring(count, len1 - count);     → O(N)
    }
    else    → O(1)
    {
        return "0"; → O(1)
    }
}
```

2. Check Function:-

```
public bool check(string str1, string str2)     → O(N)
{
    int len1 = str1.Length;     → O(1)
    int len2 = str2.Length;     → O(1)
    if (len1 > len2)    → O(1)
    {
        return false;   → O(1)
    }
    else if (len2 > len1)   → O(1)
    {
        return true;    → O(1)
    }
    else     → O(N)
    {
        for (int i = 0; i < len1; i++)      → O(N)
        {
            if (str1[i] > str2[i])      → O(1)
            {
                return false;       → O(1)
            }
            else if (str2[i] > str1[i])     → O(1)
            {
                return true;    → O(1)
            }
        }
        return false;   → O(1)
    }
}
```

3. Subtraction Function:-

```
public string Subtraction(string first, string second)      → O(N)
{
    if (check(first, second))   → O(N)
    {
        string temp = first;    → O(N)
        first = second;         → O(N)
        second = temp;          → O(N)
    }
    int count;          → O(1)
    int borrow = 0;     → O(1)
    int len1 = first.Length;    → O(1)
    int len2 = second.Length;   → O(1)
    StringBuilder sub_rusIt = new StringBuilder(first);     → O(1)
    count = len1 - len2;        → O(1)
    for (int i = len2 - 1; i >= 0; i--)         → O(N)
    {
        if ((first[i + count] - '0'+borrow) < (second[i] - '0'))        → O(1)
        {
            sub_rusIt[i + count] = (char)(((first[i + count] - '0') + 10) - (second[i] - '0') + borrow + 48); → O(1)
            borrow = -1;        → O(1)
        }
        else    → O(1)
        {
            sub_rusIt[i + count] = (char)((first[i + count] - '0') - (second[i] - '0') + borrow + 48);      → O(1)
            borrow = 0;         → O(1)
        }
    }
```

```
    for (int j = count - 1; j >= 0; j--)        → O(N)
    {
        if ((first[j] - '0' + borrow) < 0 && j != 0)        → O(1)
        {
            sub_rusIt[j] = (char)((first[j] - '0') + 10 + 48 + borrow);     → O(1)
            borrow = -1;        → O(1)
        }
        else        → O(1)
        {
            sub_rusIt[j] = (char)((first[j] - '0') + 48 + borrow);          → O(1)
            borrow = 0;         → O(1)
        }
    }
    if (sub_rusIt[0] != '0')        → O(1)
    {
        return sub_rusIt.ToString();        → O(1)
    }
    count = 0;          → O(1)
    bool finsh = false;     → O(1)
    for (int i = 0; i < len1; i++)      → O(N)
    {
        if (sub_rusIt[i] != '0')        → O(1)
        {
            finsh = true;   → O(1)
            break;          → O(1)
        }
        else        → O(1)
        {
            count++;        → O(1)
        }
    }
    if (finsh) → O(N)
    {
        return sub_rusIt.ToString().Substring(count, len1 - count);     → O(N)
    }
    else    → O(1)
    {
        return "0"; → O(1)
    }
}
```

4. Multiplication Function:-

```
public string multiplication(string first, string second)  → 3T(N/2)+O(N) → O(N^Log2(3)) → O(N^1.58)
{
    int cutpos;     → O(1)
    StringBuilder first_number = new StringBuilder(first);          → O(N)
    StringBuilder second_number = new StringBuilder(second);        → O(N)
    int count;      → O(1)
    int len1 = first_number.Length;     → O(1)
    int len2 = second_number.Length;    → O(1)

    if (len1 == 0 || len2 == 0)     → O(1)
    {
        return "0";     → O(1)
    }
    if (len1 > len2)     → O(N)
    {
        StringBuilder tmp = new StringBuilder();    → O(1)
        count = len1 - len2;                        → O(1)
        for (int i = 0; i < count; i++)             → O(N)
        {
            tmp.Append("0");                        → O(1)
        }
        tmp.Append(second);                         → O(N)
        second_number = tmp;                        → O(N)
        len2 = second_number.Length;                → O(1)
        second = second_number.ToString();          → O(N)
    }
```

```
    else if (len1 < len2)          → O(N)
    {
        StringBuilder tmp = new StringBuilder();    → O(1)
        count = len2 - len1;                    → O(1)
        for (int i = 0; i < count; i++)          → O(N)
        {
            tmp.Append("0");                    → O(1)
        }
        tmp.Append(first);                      → O(N)
        first_number = tmp;                     → O(N)
        len1 = first_number.Length;             → O(1)
        first = first_number.ToString();        → O(N)
    }

    if (len1 == 1 && len2 == 1)     → O(1)
    {
        return ((first_number[0] - '0') * (second_number[0] - '0')).ToString();     → O(1)
    }
    else    → O(1)
    {
        if (len1 % 2 == 0)          → O(1)
        {
            cutpos = len1 / 2;      → O(1)
        }
        else                        → O(1)
        {
            cutpos = len1 / 2 + 1;  → O(1)
        }

        string a = first.Substring(len1 - cutpos);              → O(N)
        string b = first.Remove(len1 - cutpos);                 → O(N)
        string c = second.Substring(len2 - cutpos);             → O(N)
        string d = second.Remove(len2 - cutpos);                → O(N)
        int l = a.Length + c.Length;                        → O(1)
        string ac = multiplication(a, c);                   → T(N/2)
        string bd = multiplication(b, d);                   → T(N/2)
        string ab_cd = multiplication(Addtion(a, b), Addtion(c, d));    → T(N/2)
        string term0 = Subtraction(Subtraction(ab_cd, ac), bd);         → O(N) + O(N)
        string term1 = term0.PadRight(term0.Length + l / 2, '0');       → O(N)
        string term2 = bd.PadRight(bd.Length + l, '0');         → O(N)
        return Addtion(Addtion(term1, term2), ac);              → O(N) + O(N)
    }
}
```

5. Division Function:-

```
public string[] Division(string first, string second)          → O(N)
{
    string[] result = new string[2];        → O(1)
    if (check(first, second))               → O(N)
    {
        result[0] = "0";        → O(1)
        result[1] = first;      → O(1)
        return result;          → O(1)
    }

    result= Division(first, Addtion(second, second));     → T(N/2)+O(N)
    result[0] = Addtion(result[0], result[0]);            → O(N)

    if (check(result[1],second))  → O(N)
    {
        return result;          → O(1)
    }
    result[0] = Addtion(result[0], "1");          → O(N)
    result[1] = Subtraction(result[1], second);    → O(N)
    return result;      → O(1)
}
```

6. ModOfPower Function:-

```
public string ModOfPower(string number, string pow, string divided)        → O((N^1.58)*Log(p))
{
   int plen = pow.Length;          → O(1)
   if (plen == 1 && pow[0] == '1')     → O(N)
   {
      return Division(number, divided)[1];     → O(N)
   }
   else if ((pow[plen - 1] - '0') % 2 == 0)    → T(P/2) + O(N^1.58)+O(P)
   {
      string number_pow = ModOfPower(number, Division(pow, "2")[0],divided);      → T(P/2) + O(P)
      return Division( multiplication(number_pow, number_pow),divided)[1];       → O(N) + O(N^1.58)
   }
   else                                                → T(P/2) + O(N^1.58)+O(P)
   {
      string number_pow = ModOfPower(number, Division(pow, "2")[0],divided);     →  T(P/2) + O(P)
                  return Division(multiplication(Division(number, divided)[1],multiplication(number_pow, number_pow)),
                divided)[1];                    → O(N) + O(N^1.58) + O(N) + O(N^1.58)
   }
}
```

7. Encryption Function:-

```
public string encryption(string number, string pow, string divided)        → O((N^1.58)*Log(p))
{
   return ModOfPower(number, pow, divided);        → O((N^1.58)*Log(p))
}
```

8. Decryption Function:-

```
public string decryption(string number, string pow, string divided)        → O((N^1.58)*Log(p))
{
   return ModOfPower(number, pow, divided);        → O((N^1.58)*Log(p))
}
```

The execution time of “Complete Test”
124062 ms = 124.062 seconds