* Assignment 4 *

Q-1 Compare Iterative and Recursive algorithm to find out Fibonacci series.

Iterative Algorithm

```
Itr_fib(int n)
{
    f[0] = f[1] = 1;
    for(i=2; i<n; i++)
    {
        f[i] = f[i-1] + f[i-2];
    }
}
```

→ The Time Complexity of iterative algo is O(n) as only one for loop is required.

→ Fast in execution

→ Iterative method makes longer code.

→ Easy to debug

Recursive Algorithm

```
Rec_fib(int n)
{
    if (n <= 2)
    {
        return 1;
    }
    else
        return (fib(n-2) + fib(n-1));
}
```

→ The Time Complexity of Recursive algo. is exponentially increasing.

→ Slow in execution

→ Reduces size of Actual code

→ Hard to debug.

Q-3 Write a program / algorithm of Selection Sort method. What is Complexity of this method?

```
Selection Sort (A)
{
    n ← length[A]
    for (j=1 ; j<n-1 ; j++)
    {
        smallest = j;
    }
    for (i=j+1 ; j<n ; i++)
    {
        if A[i] < A[smallest]
        {
            smallest = i;
        }
    }
    exchange (A[j], A[smallest])
}
```

Time Complexity:-

It 1 :- j = 0	i = 1, 2, ... n	n
It 2 :- j = 1	i = 2, ... n	n-1
It 3 = j = 2	i = 3, ... n	n-2
:	:	:
It n-1	j = n-1	i = n-1	1

Total Time :- $n + (n-1) + (n-2) + \cdots$

$$T_c = \frac{n(n+1)}{2}$$

$$\boxed{T_c = O(n^2)}$$

Ex:- 70 30 20 50 60 10 40

1. 70 30 20 50 60 10 40
   min, i; find smallest

2. 70 30 20 50 60 10 40
   min, smallest
   swap A[min] with A[smallest] ; min > smallest

3. 10 30 20 50 50 70 40
   min, j; find smallest

4. 10 20 20 50 60 70 40
   min, i smallest

5. 10 20 30 50 60 70 40
   min, j; find smallest

6. 10 20 30 50 60 70 40
   min, smallest

No swap bcz (min < smallest)

7: | 10 | 20 | 30 | 50 | 60 | 70 | 40 |
min i smallest

8. | 10 | 20 | 30 | 40 | 60 | 70 | 50 |
min i smallest

9. | 10 | 20 | 30 | 40 | 50 | 70 | 60 |
min i

10 | 10 | 20 | 30 | 40 | 50 | 60 | 70 |

Q-4 Sort the letters of word "DESIGN" in alphabetic order using bubble sort

0 1 2 3 4 5
D E S I G N

Pass - 1 :-

D E S I G N
A[0] < A[E]
D E S I G N
A[1] < A[2]
D E S I G N
A[2] > A[3] Swap
D E I S G N
A[3] > A[4] Swap
D E I G S N
A[4] > A[5] Swap
D E I G N S

Pass - 2 Array :-

Pass-2 :-

D E I G N S

D E I G N S A[0] < A[1]
A[1] < A[2]

D E I G N S A[2] ≮ A[3], swap

D E G I N S A[3] < A[4]
A[4] < A[5]

D E G I N S

Pass-3 :-

D E G I N S

A[0] < A[1]

D E G I N S

A[1] < A[2]

D E G I N S

A[3] < A[4]

D E G I N S

A[4] < A[5]

Now, in Pass-3, no swap occured, so we can say that, the array is Sorted after Pass-3.

D E G I N S.

Q-5 Using greedy algorithm find an optimal Schedual for following jobs with n=7.
Profit: $(P_1, P_2, P_3, P_4, P_5, P_6, P_7) = (3, 5, 18, 20, 6, 1, 38)$
deadline: $(d_1, d_2, d_3, d_4, d_5, d_6, d_7) = (1, 3, 3, 4, 1, 2, 1)$

Step 1:- Arrang Profit p_i in ascending order

Job	P_7	P_4	P_3	P_5	P_2	P_1	P_6
Profit	38	20	18	6	5	3	1
deadline	1	4	3	1	3	1	2

St 2 Create an array J[], which Store the job. Initially J[] will be

1	2	3	4	5	6	7
0	0	0	0	0	0	0

1	2	3	4	5	6	7	← deadline
P_7	P_2	P_3	P_4				

→ for here, P_7, P_4, P_3 will allocated Their Slots, Now at location 2, P_5 arise, but its deadline is over. then P_2 comes, it's deadline = 3 and empty "block = 2. So P_3 get its slot.

Job-Sequence :- 7-2-3-4 with 81 Profit.

Q-6 Answer the following :-

6.1 find big theta and big - omega(Ω) notation

① $f(n) = 14 * 7 + 83$

② $f(n) = 83n^3 + 84n$

6.2) Is $2n+1 = O(2n)$? Explain

Ans → 6.1)

$f(n) = 14 * 7 + 83$

$\Downarrow$

$f(n) = (7+7) * 7 + 83 = 2n^2 + 83 \; ; n = 7.$

Big-oh :- $f(n) \not\leq c \cdot g(n)$

$2n^2 + 83 \not\leq c \cdot g(n)$

Suppose $g(n) = 7(n)$

$2n^2 + 83 \not\leq 7(n)$ for $n \neq 7$

$\boxed{2n^2 + 83 = \Omega(n)}$

big-Theta

$c_1 \cdot g(n) \leq f(n) \leq c_2 * g(n)$

in earlier case we suppose $g(n) = f(n)$

6.9

$7n \leq 2n^2 + 83 \leq 50n$

$c_1 = 7, \quad c_2 = 50 \quad$ for $n = 7$.

$\boxed{f(n) = \Theta(n)}$

ii) $f(n) = 83n^3 + 84n$

① Big-Ω Notation:-

$f(n) \geq c \cdot g(n)$

$83n^3 + 84n > n^2 \quad$ for all $n > 0$.

$\boxed{f(n) = \Omega(n^2)}$

② Big-Theta Notation:-

$c_1 g(n) \leq f(n) \leq c_2 g(n)$

$c_1 n^3 \leq 83n^3 + 84n \leq c_2 n^3$

$c_1 = 83 \qquad c_2 = 167$

$\boxed{f(n) = \Theta(n^3)}$

6.2 $2^{n+1} = O(2^n)$

Ex $2^{n+1} \leq c \cdot 2^n$ for $c \geq 2$

So $\boxed{2^{n+1} = O(2^n)}$ True.

Q-7 Find big-oh (O) notation for following :-
1) $f(n) = 6993$ 2) $f(n) = 6n^2 + 135$

→ Big-oh (O) Notation :-

$f(n) = O(g(n)) \Rightarrow f(n) \ll (g(n)$

i) $f(n) = 6993$. (Consider 6993 as n)
for $c = 1$, $g(n) = 6993$.

$\boxed{f(n) = O(n)}$

ii) $f(n) = 6n^2 + 135$

Suppose $g(n) = n^2$

$f(n) \leq c \cdot g(n)$

~~let 6 and~~

$6n^2 + 135 \leq 141 \, n^2$ $c = 141$
for all $n > 0$

$\boxed{f(n) = O(n^2)}$