Answer Script

Question No. 01

Problem - 1

Link: Fibonacci Number - LeetCode **20**

Answer No. 01

Code:

Memoization:

```
#include<bits/stdc++.h>
using namespace std;
const int N = 101;
int dp[N];
int fib(int n)
{
    if(n <= 2)
        if(n==0)
            return 0;
        else
            return 1;

    if(dp[n] != -1)
        return dp[n];

    int ans = fib(n-1) + fib(n-2);

    dp[n] = ans;
    return ans;
}
int main()
{
    int n;
    cin>>n;

    for(int i=1; i<=n; i++)
        dp[i] = -1;
    cout<<fib(n)<<"\n";
    return 0;
}
```

Tabulation:

```
#include<bits/stdc++.h>
using namespace std;

const int N = 101;
int dp[N];

int main()
{
   int n;
   cin>>n;

   dp[1] = 1;
   dp[2] = 1;

   for(int i=3; i<=n; i++)
   {
      dp[i] = dp[i-1] + dp[i-2];
   }

   cout<<dp[n]<<"\n";

   return 0;
}
```

Question No. 02

Problem - 2

Link: FARIDA **20**

Answer No. 02

Code:

Memoization:

```
#include<bits/stdc++.h>
using namespace std;
#define ll long long int

const ll N = 1e5;
ll v[N], dp[N];

ll solve(int n)
{
   if (n == 0)
   {
      return 0;
   }

   if (dp[n] != -1)
   {
      return dp[n];
   }

   ll ans = max(v[n] + solve(n-2), solve(n-1));

   dp[n] = ans;
   return ans;
}

int main()
{
   int t;
   cin>>t;

   for(int i=1; i<=t; i++)
```

```
    {
        int n;
        cin>>n;

        for(int j=1; j<=n; j++)
            cin>>v[j];

        for(int j=1; j<=n; j++)
            dp[j] = -1;

        cout<<"Case "<<i<<": "<<solve(n)<<"\n";
    }
    return 0;
}
```

Tabulation:

```
#include<bits/stdc++.h>
using namespace std;
#define ll long long int

int main()
{
    int t;
    cin>>t;

    for(int i=1; i<=t; i++)
    {
        int n;
        cin>>n;

        if(n==0)
        {
            cout<<"Case "<<i<<": 0"<<"\n";
            continue;
        }

        vector<ll> v(n);
        vector<ll> dp(n);

        for(int i=0; i<n; i++)
            cin>>v[i];

        dp[0] = v[0];
```

```
        dp[1] = max(dp[0], v[1]);

        for(int j=2; j<n; j++)
        {
            dp[j] = max(dp[j-1], dp[j-2] + v[j]);
        }

        cout<<"Case "<<i<<": "<<dp[n-1]<<"\n";
    }
    return 0;
}
```

Question No. 03

Problem - 3

Link: Boredom **20**

Answer No. 03

Code:

Memoization:

```
#include<bits/stdc++.h>
using namespace std;
#define ll long long int
const ll N = 1e6 + 7;
ll dp[N], temp[N];

ll solve(ll n)
{
   if(n==0)
      return 0;

   if(n==1)
      return dp[1];

   if(temp[n] != -1)
      return temp[n];

   ll ans = max(solve(n-1), solve(n-2) + n*dp[n]);
   temp[n] = ans;
   return ans;
}

int main()
{
   ll n, x, y = 0;
   cin>>n;

   for(ll i=0; i<N; i++)
      temp[i] = -1;

   for(ll i=0; i<n; i++)
```

```
    {
        cin>>x;
        dp[x]++;
        y = max(y, x);
    }

    cout<<solve(y)<<"\n";

    return 0;
}
```

Tabulation:

```
#include<bits/stdc++.h>
using namespace std;
#define ll long long int
const ll N = 1e5 + 5;
ll dp[N];
ll dp2[N];

int main()
{
    ll n, x;
    cin>>n;

    for(int i=0; i<n; i++)
    {
        cin>>x;
        dp[x]++;
    }

    dp2[0] = 0;
    dp2[1] = dp[1];

    for(int i=2; i<=N; i++)
    {
        dp2[i] = max(dp2[i-1], dp2[i-2] + i*dp[i]);
    }

    cout<<dp2[N]<<"\n";

    return 0;
}
```

Question No. 04

Problem - 4

Link: N-th Tribonacci Number - LeetCode **20**

Answer No. 04

Code:

Memoization:

```
#include<bits/stdc++.h>
using namespace std;

const int N = 101;
int dp[N];

int tribonacci(int n)
{
   if(n<=2)
      if(n==0)
         return 0;
      else
         return 1;

   if(dp[n] != -1)
      return dp[n];

   int ans = tribonacci(n-1) + tribonacci(n-2) + tribonacci(n-3);

   dp[n] = ans;
   return ans;
}

int main()
{
   int n;
   cin>>n;

   for(int i=1; i<=n; i++)
      dp[i] = -1;
```

```
    cout<<tribonacci(n)<<"\n";

    return 0;
}
```

Tabulation:

```
#include<bits/stdc++.h>
using namespace std;

const int N = 101;
int dp[N];

int main()
{
    int n;
    cin>>n;

    dp[1] = 1;
    dp[2] = 1;

    for(int i=3; i<=n; i++)
        dp[i] = dp[i-1] + dp[i-2] + dp[i-3];

    cout<<dp[n]<<"\n";

    return 0;
}
```

Question No. 05

Problem - 5

➔ You are given an integer n.You can perform any of the following operations on it as many times you want -

- Subtract 1 from it
- If it is divisible by 2 divide by 2
- If it is divisible by 3 divide by 3

Find the minimum number of operations to make n=1 **20**

Constraints -

1<=n<=10^5

Output -

Print a single integer,the minimum number of operations to make n=1

Sample Input-	**Sample Output-**
7	3
11	4

Explanation-

When n = 7,
By using 3 operations we can go from 7 to 1.
>> 1st step -> subtract 1 from 7 then it became 6
>> 2nd step -> 6 is divisible by 3 hence we can divide it by 3 and it became 2
>> 3rd step -> 2 is divisible by 2 hence we can divide it by 2 and it became 1

Answer No. 05

Code:

Memoization:

```
#include<bits/stdc++.h>
using namespace std;

const int N = 1e5+5;
const int INF = 2e9;
int dp[N];

int solve(int n, int *dp)
```

```
{
    if(n<=1)
        return 0;

    if(dp[n] != -1)
        return dp[n];

    int a = solve(n-1, dp);
    int b = INF;
    int c = INF;

    if(n%2==0)
        b = solve(n/2, dp);
    if(n%3==0)
        c = solve(n/3, dp);

    int ans = 1 + min(a, min(b,c));

    dp[n] = ans;
    return dp[n];
}

int main()
{
    int n;
    cin>>n;

    for(int i=1; i<=n; i++)
        dp[i] = -1;

    cout<<solve(n, dp)<<"\n";

    return 0;
}
```

Tabulation:

```
#include<bits/stdc++.h>
using namespace std;

const int N = 1e5 + 5;
const int INF = 2e9;
int dp[N];

int main()
{
   int n;
   cin>>n;

   for(int i=0; i<n; i++)
      dp[i] = -1;

   dp[1] = 0;

   int a, b, c;
   for(int i=2; i<=n; i++)
   {
      a = dp[i-1];
      b = INF;
      c = INF;

      if(i%2==0)
         b = dp[i/2];
      if(i%3==0)
         c = dp[i/3];

      dp[i] = 1 + min(a, min(b,c));
   }

   cout<<dp[n]<<"\n";

   return 0;
}
```