Answer Script

Question No. 01	
Fibonacci Number	**20**

Answer No. 01

Memoization Method:

```
#include<bits/stdc++.h>
using namespace std;
const int N = 101;

int dp[N];

int fib(int n)
{
        if(n <= 2) 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) << endl;
        return 0;
}
```

Tabulation Method:

```
#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] << endl;
        return 0;
}
```

Question No. 02	
FARIDA	**20**

Answer No. 02

Memoization Method:

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

vector<long long int> dp(1000000);

long long int maxloot(vector<int> &loot, int n)
{
        if (n == 0)
        return 0;
        if (n == 1)
        return loot[0];
        if (n == 2)
        return max(loot[0], loot[1]);
        if (dp[n] != -1)
        return dp[n];
        long long int take = loot[n - 1] + maxloot(loot, n - 2);
        long long int leave = maxloot(loot, n - 1);
        return dp[n] = max(take, leave);
}

int main()
{
        int t;
        cin >> t;
        int p = 1;
        while (t--)
        {
        int n;
        cin >> n;
        vector<int> loot;
        for (int i = 0; i < n; i++)
```

```
        {
        int temp;
        cin >> temp;
        loot.push_back(temp);
        }
        dp.assign(1000000, -1);

        cout << "Case " << p << ": " << maxloot(loot, n) << endl;
        p++;
        dp.clear();
        }
}
```

Tabulation Method:

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

int main()
{
        int T, N, c;
        long long tmp, dp[4];

        cin >> T;
        for(int tc=1; tc <= T; ++tc) {
        cin >> N;
        c = 0;
        for(int i=0; i<4; ++i)
        dp[i] = 0;
        for(int i=0; i<N; ++i) {
        cin >> tmp;
        dp[c] = tmp + max( dp[(c+2)%4], dp[(c+1)%4] );
        c = (c+1)%4;
        }
        tmp = max( dp[(c+2)%4], dp[(c+3)%4] );
        cout << "Case " << tc << ": " << tmp << "\n";
        }
        return 0;
}
```

Question No. 03	
Boredom	**20**

Answer No. 03

Memoization Method:

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

int n;
map<int, int> m;
vector<pair<int, int>> v;
long long f[111111];

int nextInt()
{
	int x = 0, p = 1;
	char c;
	do {
	c = getchar();
	} while (c <= 32);
	if (c == '-') {
	p = -1;
	c = getchar();
	}
	while (c >= '0' && c <= '9') {
	x = x * 10 + c - '0';
	c = getchar();
	}
	return x * p;
}

long long solve(int idx)
{
	if (idx == 0) return 1LL * v[idx].first * v[idx].second;
	if (f[idx] != -1) return f[idx];

	int pr = idx - 1;
```

```
        while (pr >= 0 && v[pr].first + 1 == v[idx].first) pr--;

        if (pr == -1)
        f[idx] = 1LL * v[idx].first * v[idx].second;
        else
        f[idx] = solve(pr) + 1LL * v[idx].first * v[idx].second;

        f[idx] = max(f[idx], solve(idx - 1));
        return f[idx];
}

int main()
{
        n = nextInt();
        while (n--) {
        int x = nextInt();
        m[x]++;
        }
        for (map<int, int>::iterator it = m.begin(); it != m.end(); it++) {
        v.push_back(make_pair(it->first, it->second));
        }

        memset(f, -1, sizeof(f));
        cout << solve(v.size() - 1) << "\n";

        return 0;
}
```

Tabulation Method:

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

int n;
map<int, int> m;
vector<pair<int, int>> v;
long long f[111111];

int nextInt()
{
```

```
        int x = 0, p = 1;
        char c;
        do {
        c = getchar();
        } while (c <= 32);
        if (c == '-') {
        p = -1;
        c = getchar();
        }
        while (c >= '0' && c <= '9') {
        x = x * 10 + c - '0';
        c = getchar();
        }
        return x * p;
}

int main()
{
        n = nextInt();
        while (n--) {
        int x = nextInt();
        m[x]++;
        }
        for (map<int, int>::iterator it = m.begin(); it != m.end(); it++) {
        v.push_back(make_pair(it->first, it->second));
        }

        f[0] = 1LL * v[0].first * v[0].second;

        for (int i = 1; i < v.size(); i++) {
        int pr = i - 1;
        while (pr >= 0 && v[pr].first + 1 == v[i].first) pr--;

        if (pr == -1)
        f[i] = 1LL * v[i].first * v[i].second;
        else
        f[i] = f[pr] + 1LL * v[i].first * v[i].second;

        f[i] = max(f[i], f[i - 1]);
        }
```

```
        cout << f[v.size() - 1] << "\n";

        return 0;
}
```

Question No. 04	
N-th Tribonacci Number	**20**

Answer No. 04

Memoization Method:

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

vector<int> dp(101, -1);
int tribonacciMemo(int n)
{
	if (n <= 0)
	return 0;
	if (n == 1 || n == 2)
	return 1;
	if (dp[n] != -1)
	return dp[n];
	dp[n] = tribonacciMemo(n - 1) + tribonacciMemo(n - 2) + tribonacciMemo(n
- 3);
	return dp[n];
}

int main()
{
	int n;
	cin >> n;
	int tribN = tribonacciMemo(n);
	cout << tribN << "\n";
	return 0;
}
```

Tabulation Method:

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

int tribonacciTab(int n)
```

```
{
        vector<int> dp(n + 1);
        dp[0] = 0;
        dp[1] = 1;
        dp[2] = 1;
        for (int i = 3; i <= n; i++)
        {
        dp[i] = dp[i - 1] + dp[i - 2] + dp[i - 3];
        }
        return dp[n];
}

int main()
{
        int n;
        cin >> n;
        int tribN = tribonacciTab(n);
        cout << tribN << "\n";
        return 0;
}
```

Question No. 05

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

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

Memoization Method:

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

unordered_map<int, int> dp;

int minOperations(int n) {
        if (n == 1) {
        return 0;
        }
        if (dp.find(n) != dp.end()) {
        return dp[n];
        }

        int ans = 1 + minOperations(n - 1);
        if (n % 2 == 0) {
        ans = min(ans, 1 + minOperations(n / 2));
        }
        if (n % 3 == 0) {
        ans = min(ans, 1 + minOperations(n / 3));
        }

        dp[n] = ans;
        return ans;
}

int main() {
        int n;
        cin >> n;
        int ans = minOperations(n);
        cout << ans << "\n";
        return 0;
}
```

Tabulation Method:

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

int minOperations(int n) {
```

```
        vector<int> dp(n + 1, 0);

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

        if (i % 2 == 0) {
        dp[i] = min(dp[i], 1 + dp[i / 2]);
        }

        if (i % 3 == 0) {
        dp[i] = min(dp[i], 1 + dp[i / 3]);
        }
        }
        return dp[n];
}

int main() {
        int n;
        cin >> n;
        int ans = minOperations(n);
        cout << ans << "\n";
        return 0;
}
```