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

struct Process
{
    int id;
    int size;
    int allocation;
    bool isGiven = false;
};

struct Memory
{
    int size;
    int free;
    int allocated;
    bool isTaken = false;
    int extfrag;
    int givenProcessId = -1;
};

int m;
int n;
int external_fragmentation = 0;
int internal_fragmentation = 0;

void firstFit(Process p[], int n, Memory mem[], int m)
{

    // pick each process and find suitable blocks
    // according to its size ad assign to it
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < m; j++)
        {
            if (mem[j].size >= p[i].size)
            {
                mem[j].isTaken = true;
                p[i].isGiven = true;
                mem[j].givenProcessId = p[i].id;
                mem[j].free -= p[i].size;
                mem[j].size -= p[i].size;
                p[i].allocation = j + 1;
                mem[j].allocated = p[i].id;
                break;
            }
        }
    }
}

void calcfrag(Process p[], int n, Memory mem[], int m)
{
    int flag = 0;
    for (int i = 0; i < n; i++)
    {
        if (p[i].isGiven != true)
        {
            flag = 1;
            break;
        }
    }
    if (flag == 0)
    {
        external_fragmentation = 0;
    }
    else
    {
        for (int i = 0; i < m; i++)
        {
            if (mem[i].isTaken != true)
            {
                external_fragmentation += mem[i].size;
            }
        }
    }

    for (int i = 0; i < m; i++)
    {
        if (mem[i].isTaken != false)
        {
            internal_fragmentation += mem[i].free;
        }
    }
}

void printTable(Process P[], int n, Memory mem[], int m, int memorySize[])
{

    for(int i=0;i<m;i++)
    {
        if(mem[i].free==memorySize[i])
        {
               mem[i].free=0;
        }
    }

    cout << "\nTable-->(-1 Denotes Unallocated process)\n";
    int i;

    puts("+-----+------------+--------------+-----------------+");
    puts("| BNO | Block Size | Process All. | Internal Fragm. |");
    puts("+-----+------------+--------------+-----------------+");

    for (i = 0; i < m; i++)
    {
        printf("| %2d  |     %2d    |      %2d      |        %3d      |\n", i, memorySize[i], mem[i].givenProcessId, mem[i].free);
        puts("+-----+------------+--------------+-----------------+");
    }

    cout << "External Fragmentation: " << external_fragmentation << endl;
    cout << "Internal Fragmentation: " << internal_fragmentation << endl;
}

int main()
{
    cout << "\nEnter the number of memory blocks: ";
    cin >> m;
    Memory mem[m];
    int memorySize[m];
    for (int i = 0; i < m; i++)
    {
        cout << "\n";
        cout << "Enter the size of the memory block " << i + 1 << ": ";
        cin >> mem[i].size;
        mem[i].free = mem[i].size;
        mem[i].allocated = -1;
        mem[i].extfrag = 0;
        memorySize[i] = mem[i].size;
    }

    cout << "\nEnter the number of processes: ";
    cin >> n;
    Process p[n];
    for (int i = 0; i < n; i++)
    {
        p[i].id = i + 1;
        cout << "\n";
        cout << "\nEnter the size of the process" << p[i].id << ": ";
        cin >> p[i].size;
    }

    firstFit(p, n, mem, m);

    calcfrag(p, n, mem, m);

    printTable(p, n, mem, m, memorySize);
}
```