#include<bits/stdc++.h>

using namespace std;

// siz of vector of pairs

int siz;

// Global vector of pairs to store

// address ranges available in free list

vector<pair<int, int>> free\_list[100000];

// Map used as hash map to store the starting

// address as key and siz of allocated segment

// key as value

map<int, int> mp;

void initialize(int sz)

{

// Maximum number of powers of 2 possible

int n = ceil(log(sz) / log(2));

siz = n + 1;

for(int i = 0; i <= n; i++)

free\_list[i].clear();

// Initially whole block of specified

// siz is available

free\_list[n].push\_back(make\_pair(0, sz - 1));

}

void allocate(int sz)

{

// Calculate index in free list

// to search for block if available

int n = ceil(log(sz) / log(2));

// Block available

if (free\_list[n].size() > 0)

{

pair<int, int> temp = free\_list[n][0];

// Remove block from free list

free\_list[n].erase(free\_list[n].begin());

cout << "Memory from " << temp.first

<< " to " << temp.second << " allocated"

<< "\n";

// map starting address with

// siz to make deallocating easy

mp[temp.first] = temp.second -

temp.first + 1;

}

else

{

int i;

for(i = n + 1; i < siz; i++)

{

// Find block siz greater than request

if(free\_list[i].size() != 0)

break;

}

// If no such block is found

// i.e., no memory block available

if (i == siz)

{

cout << "Sorry, failed to allocate memory \n";

}

// If found

else

{

pair<int, int> temp;

temp = free\_list[i][0];

// Remove first block to split it into halves

free\_list[i].erase(free\_list[i].begin());

i--;

for(; i >= n; i--)

{

// Divide block into two halves

pair<int, int> pair1, pair2;

pair1 = make\_pair(temp.first,

temp.first +

(temp.second -

temp.first) / 2);

pair2 = make\_pair(temp.first +

(temp.second -

temp.first + 1) / 2,

temp.second);

free\_list[i].push\_back(pair1);

// Push them in free list

free\_list[i].push\_back(pair2);

temp = free\_list[i][0];

// Remove first free block to

// further split

free\_list[i].erase(free\_list[i].begin());

}

cout << "Memory from " << temp.first

<< " to " << temp.second

<< " allocated" << "\n";

mp[temp.first] = temp.second -

temp.first + 1;

}

}

}

// Driver code

int main()

{

int total,c,req;

printf("Enter the total size of memory: ");

cin>>total;

initialize(total);

printf("Enter the no. of processes: ");

cin>>c;

while(c>0)

{

printf("Enter the size of process: ");

cin>>req;

if(req < 0)

break;

allocate(req);

c-=1;

}

// initialize(128);

// allocate(32);

// allocate(7);

// allocate(64);

// allocate(56);

return 0;

}