```
/*
This program iterates through a file of logical addresses and converts the logical
addresss to the physical address.
To do this we create a paging table to map the virtual address to page number and
offset.
Formula :
PageNumber = virtual_address / page_size
PageOffset = virtual_address % page_size
FrameNumber = (number between 1 & 256) / page_size
PhysicalAddress = (FrameNumber * page_size) + PageOffset
*/

#include <iostream>
#include <stdlib.h>
#include <fstream>
#include <string>
#include <random>
#include <cmath>


//Global Variables
int n = 0;
int n1 = 0;
int* virtual_addresses1;
int* virtual_addresses2;
int* page_number;
int* page_offset;
int* physical_address;

//Page table
struct table{
    int frame_number;

    table(int x = 0) : frame_number(x){}
};

//Global Constants
int num_page_entries = 256;
int page_size = pow(2,8);
int frame_size = pow(2,8);
int number_of_frames = 256;
using namespace std;
ifstream inFile;
ifstream inFile1;

void readData(){
    //Reads text file of logical addresses and overwrites the initliazed array
    int virtual_address;
    string path2file =
"C:\\Users\\zuck1\\C_Projects\\Norwich\\CS270A\\addresses.txt";

    int i;
    int index = 0;

    inFile.open(path2file);
    inFile1.open(path2file);
    //To handle an array of different sizes from file inputs we count the lines to
set the size

    while(inFile >> virtual_address){
        //cout << virtual_address << "\n";
        n++;
    }

    //We then allocate memory to the count of the lines in the file setting the
array to size n
    virtual_addresses1 = (int*) malloc(n * sizeof(int));
    cout << "There are " << n << " virtual addresses in the file" << "\n";

    //The array is then overwritten with values from the file
    while(inFile1 >> i){

        virtual_addresses1[index] = i;
        index++;
    }
}

void calculatePhysicalAddress(){
    //We use the logical address to physical address translation formulas to find a
physical address
    //Here we allocate uninitialized arrays with calloc to size n as we already know
the size we will be working with
    page_number = (int*) calloc(n , sizeof(int));
    page_offset = (int*) calloc(n , sizeof(int));
    physical_address = (int*) calloc(n , sizeof(int));
    table* page_table = new table[page_size];

    for(int i = 0; i < n; i++){
        //To find the page number of the virtual address within the page table we
divide the virtual address by the page size
        //To find the page off set we take the modulo of the virtual address by the
size
        page_number[i] = virtual_addresses1[i]/page_size;
        page_offset[i] = virtual_addresses1[i] % page_size;
        //cout << page_offset[i]  << "for virtual address " << virtual_addresses1[i]
<< "\n";
        //cout << page_number[i] << "  page number " << "\n";
    }
    cout << "\n\n";
    for(int i = 0; i < n; i++){
        //To create a mapping between the page table and the frame number we set
each index of the page table to a frame number between 1 and 256
        page_table[i].frame_number = (int)rand() % page_size;

    }

    for(int i = 0; i < n; i++){
        //We then calculate a physical address by multiplying the frame number by
the page size and add the offset
        physical_address[i] = (page_table[i].frame_number * page_size) +
page_offset[i];

        //cout << "Virtual Address : "<< virtual_addresses1[i] << " " << "Physical
Address: " << physical_address[i] << "\n";
```

```
    }
}

void print_physical_address(){
    //To demonstrate the deallocation of memory and print a smaller array we resize
the virtual and physical arrays
    int new_size = 25;

    virtual_addresses1 = (int*)realloc(virtual_addresses1,new_size * sizeof(int));
    physical_address = (int*)realloc(physical_address, new_size * sizeof(int));

    for(int i = 0; i < new_size; i++){

        cout << "Virtual Address " << "#" << i << ": " << virtual_addresses1[i] << "
" << "Physical Address: " << physical_address[i] << "\n";
    }
    //The memory is then further deallocated and returned to the system
    free(virtual_addresses1);
    free(physical_address);
}


int main(){
    readData();
    calculatePhysicalAddress();
    print_physical_address();
}
```