```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define PAGE_SIZE 256
#define PHYSICAL MEM SIZE 1024
#define LOGICAL MEM SIZE 2048
#define NUM FRAMES (PHYSICAL MEM SIZE / PAGE SIZE)
#define NUM_PAGES (LOGICAL_MEM_SIZE / PAGE_SIZE)
int physical_memory[PHYSICAL_MEM_SIZE];
int page_table[NUM_PAGES];
void initialize_page_table() {
  for (int i = 0; i < NUM_PAGES; i++) {</pre>
   page_table[i] = -1;
  }
}
int allocate_frame(int page_number) {
  int frame number = rand() % NUM FRAMES;
  page_table[page_number] = frame_number;
  return frame number;
int translate_address(int logical_address) {
  int page number = logical address / PAGE SIZE;
  int offset = logical_address % PAGE_SIZE;
  int frame_number = page_table[page_number];
  if (frame number == -1) {
    frame_number = allocate_frame(page_number);
  return frame_number * PAGE_SIZE + offset;
void write_memory(int logical_address, int value) {
  int physical address = translate address(logical address);
  physical_memory[physical_address] = value;
int read_memory(int logical_address) {
  int physical_address = translate_address(logical_address);
  return physical memory[physical address];
}
int main() {
  srand(time(0));
  initialize_page_table();
 write memory(500, 42);
 write_memory(1200, 84);
  printf("Value \ at \ logical \ address \ 500: \ %d\n", \ read\_memory(500));
  printf("Value at logical address 1200: %d\n", read_memory(1200));
  return 0;
}
```