#include<sstream>

#include<iostream>

#include<string.h>

#include<string>

#include<fstream>

#include<string>

#include<cstdlib>

using namespace std;

bool occupied\_pages[30];

int s\_to\_i(string operand)

{

if(operand[0]>='0' && operand[0]<='9' && operand[1]>='0' && operand[1]<='9')

return ((int)operand[0]-48)\*10+((int)operand[1]-48);

return -1;

}

class memory

{

private:

char mem[300][4];

char ch;

int page\_table\_ptr;

public:

void reset()

{

memset(mem,'$',sizeof(char)\*300\*4);

memset(occupied\_pages,false,sizeof(bool)\*30);

page\_table\_ptr=rand()%30;

printf("%d\n",page\_table\_ptr);

occupied\_pages[page\_table\_ptr]=true;

page\_table\_ptr\*=10;

printf("%d\n",page\_table\_ptr);

}

string get\_mem(int pos)

{

string temp="";

for(int i=0;i<4;i++)

temp+=mem[pos][i];

//cout<<"String "<<temp<<endl;

return temp;

}

void set\_mem(string s, int pos)

{

for(int i=0;i<4;i++)

mem[pos][i]=s[i];

}

int get\_page\_table\_ptr()

{

return page\_table\_ptr;

}

int allocate\_page()

{

int page\_no=rand()%30;

while(occupied\_pages[page\_no]==true)

page\_no=rand()%30;

occupied\_pages[page\_no]=true;

cout<<"Page no: "<<page\_no<<endl;

return page\_no;

}

void set\_page\_table(int row\_num,int page\_no)

{

ostringstream temp;

temp << page\_no;

string table\_entry;

if(page\_no<10)

table\_entry="\*\*0"+temp.str();

else

table\_entry="\*\*"+temp.str();

set\_mem(table\_entry,page\_table\_ptr+row\_num);

}

// void store\_card(string s,int mem\_cnt)

// {

// string word="";

// int page\_no=allocate\_page();

// printf("%d\n",page\_no);

// set\_page\_table(mem\_cnt, page\_no);

// page\_no\*=10;

// for(int i=0;i<s.length();i+=4)

// {

// for(int j=0;j<4;j++)

// {

// word+=s[i+j];

// }

// set\_mem(word,page\_no);

// page\_no++;

// //cout<<"Word: "<<word<<endl;

// word="";

// }

// }

void print\_mem()

{

for(int i=0;i<300;i++)

{

string temp;

cout<<"Data at mem location"<<"["<<i<<"]";

for(int j=0;j<4;j++)

{

cout<<mem[i][j];

}

cout<<"\n";

}

}

}m\_obj;

class cpu

{

public:

int address\_tranlation(int virtual\_add)

{

int page=m\_obj.get\_page\_table\_ptr()+(virtual\_add/10);

string value\_page=m\_obj.get\_mem(page);

value\_page=value\_page.substr(2,2);

return (s\_to\_i(value\_page)\*10+(virtual\_add%10));

}

}exe;

void GD(string s,int loc){

int pos;

int page\_no=m\_obj.allocate\_page();

m\_obj.set\_page\_table((loc)/10,page\_no);

pos=exe.address\_tranlation(loc);

pos=(pos/10)\*10;

int len=s.length(),start=0,i;

string s1;

for(i=pos;start<len;i++)

{

if((len-start)<4)

s1=s.substr(start,(len-start));

else

s1=s.substr(start,4);

start+=4;

m\_obj.set\_mem(s1,i);

}

}

int main()

{

std::string s;

m\_obj.reset();

std::cout << "Enter the data: ";

std::getline(std::cin, s);

int loc;

std::cout << "Enter the location: ";

std::cin >> loc;

GD(s,loc);

m\_obj.print\_mem();

return 0;

}