```
//============================================================================
// Name        : MorseCode.cpp
// Author      :
// Version     :
// Copyright   : Your copyright notice
// Description : LED Display, Ansi-style
//============================================================================

//Thomas Gibbons
//Oct 27, 2016
#include <iostream>
#include <sstream>
#include <string>
using namespace std;

#include <stdio.h>
#include <sys/mman.h>
#include <stdlib.h>
#include <sys/types.h>
#include <fcntl.h>
#include <unistd.h>

//-----------------------------------------------------------------
string morseChart[]=  { ".-","-...","-.-.","-..",".","..-.",
                        "--.","....","..",".---","-.-",".-..",
                        "--","-.","---",".--.","--.-",".-.",
                        "...","-","..-","...-",".--","-..-",
                        "-.--","--..","?"};
//-----------------------------------------------------------------
class message{
    private:

    protected:
        string english;
    public:
        message();
        message(string);
        ~message();
        virtual void printInfo();
};

message::message(){
    //std::cout <<  "\nObject of class message created";
    std::cout   <<  "English message to display: ";
    std::cin    >>  english;
}

message::message(string word){
    //std::cout <<  "\nObject of class message created";
    english=word;
}

message::~message(){
```

```
    //nothing needs to be freed
}

void message::printInfo(){
    std::cout   <<  "\nEnglish message: "    << english;
}

class morseCodeMessage:public message{
    private:
        string *morse;
        void translate();
        int index;
    public:
        morseCodeMessage();
        morseCodeMessage(string);
        ~morseCodeMessage();
        morseCodeMessage* next;
        int morse2Led(char);
        void printInfo();       //new standard
};

morseCodeMessage::morseCodeMessage(string x):message(x){
    translate();
    next=NULL;
}

morseCodeMessage::morseCodeMessage():message(){
    translate();
    next=NULL;
}

morseCodeMessage::~morseCodeMessage(){
    //cout << "\nGoodbye Message: ";
    delete[] morse;
}
int morseCodeMessage::morse2Led(char a){
    int fd;     // for the file descriptor of the special file we need to open.
        unsigned long *BasePtr;     // base pointer, for the beginning of the memory page
        (mmap)
        unsigned long *PBDR, *PBDDR;    // pointers for port B DR/DDR

        fd = open("/dev/mem", O_RDWR|O_SYNC);   // open the special file /dem/mem
        if(fd == -1){
            printf("\n error\n");
            return(-1);  // failed open
        }

        // We need to map Address 0x80840000 (beginning of the page)
        BasePtr = (unsigned long*)mmap(NULL,4096,PROT_READ|PROT_WRITE,MAP_SHARED,fd,
        0x80840000);
            if(BasePtr == MAP_FAILED){
                printf("\n Unable to map memory space \n");
                return(-2);
```

```
            }  // failed mmap

        // To access other registers in the page, we need to offset the base pointer to
        reach the
            // corresponding addresses. Those can be found in the board's manual.
            PBDR = BasePtr + 1;     // Address of port B DR is 0x80840004
            PBDDR = BasePtr + 5;    // Address of port B DDR is 0x80840014

    if(a == '.'){//red
        *PBDDR |= 0x20;
        //*PBDDR &= 0xFFFFFFFC;

        *PBDR |= 0x20;  // ON: write a 1 to port B0. Mask all other bits.
        sleep(1);   // How can you sleep for less than a second?
        *PBDR &= ~0x20; // OFF: write a 0 to port B0. Mask all other bits.
        sleep(1);

    }
    else if(a == '-'){//yellow
        *PBDDR |= 0x40;
        //*PBDDR &= 0xFFFFFFFD;

        *PBDR |= 0x40;  // ON: write a 1 to port B0. Mask all other bits.
        sleep(1);   // How can you sleep for less than a second?
        *PBDR &= ~0x40; // OFF: write a 0 to port B0. Mask all other bits.
        sleep(1);
    }
    else{//green
        *PBDDR |= 0x80;
        //*PBDDR &= 0xFFFFFFFE;

        *PBDR |= 0x80;  // ON: write a 1 to port B0. Mask all other bits.
        sleep(1);   // How can you sleep for less than a second?
        *PBDR &= ~0x80; // OFF: write a 0 to port B0. Mask all other bits.
        sleep(1);
    }

    close(fd);
    return 0;
}

void morseCodeMessage::printInfo(){
    cout    <<  english <<  " = ";  //same as in message class
    int count=english.length();     //length of word
    int x=0;

    while(x<count)
    {
        cout    << morse[x] <<  " ";    //prints out each morse string letter by letter
        int length=morse[x].length();
        int y=0;
        while(y<length) {
            morse2Led(morse[x][y]);
```

```
            y++;
        }
        x++;
    }
    morse2Led('!');
    cout << endl;
}

void morseCodeMessage::translate(){
    int count=english.length(); //get length of word
    int x=0;
    morse=new string[count];    //make string for each letter of word

    while(x<count)
    {
        if (isupper(english[x]))
            english[x]=tolower(english[x]);     //convert everything to lower case
        index = english[x] - 'a';   //Use ascii character 'a' to center indexes from 0 to 25
        if (index<0 || index>25)
            index=26;
        morse[x]=morseChart[index];     //find corresponding morse string for letter
        x++;
    }
}

class messageStack{
    private:
        morseCodeMessage* stack_top;
        int numObjects;
    public:
        messageStack();
        messageStack(morseCodeMessage current_obj);
        void push(morseCodeMessage* current_obj);
        morseCodeMessage pop();
        void printStack();
};

messageStack::messageStack(){
    //initialize stack
    numObjects=0;
    stack_top=NULL;
}

messageStack::messageStack(morseCodeMessage current_obj){
    //initialize stack with first object
    numObjects=1;
    stack_top=&current_obj;
}

void messageStack::push(morseCodeMessage* current_obj){
    (*current_obj).next=stack_top;  //link two objects
    stack_top=current_obj;  //keep track of top of linked list
    numObjects++;   //keep track of objects linked
```

```
}

morseCodeMessage messageStack::pop(){
    morseCodeMessage* hold=stack_top;   //hold top to return
    stack_top=(*stack_top).next;    //change top to new top
    numObjects--;   //one less object in linked list
    return *hold;   //return old top for print
}

void messageStack::printStack(){
    while(numObjects>0){
        pop().printInfo();  //get the message on top and print
    }
}

int main(int argc, char** argv){
    //make sure their are arguments other than program name
    /*if(argc==1){
        cout << "\nError: Input arguments to be converted to morse code";
        return 0;
    }

    //push arguments to stack except program name
    int count=argc;
    morseCodeMessage** hold= new morseCodeMessage*[argc];
    messageStack order;
    while(count>1){
        hold[count-1]=new morseCodeMessage(argv[count-1]);

        //(*hold).printInfo();
        order.push(hold[count-1]);
        count--;
    }

    //print stack
    order.printStack();

    //free allocated objects
    count=argc;
    while(count>1){
        delete hold[count-1];
        count--;
    }

    //free allocated pointer of array of objects
    delete[] hold;*/
    morseCodeMessage word1("Thomas");
    morseCodeMessage word2("Patrick");
    morseCodeMessage word3("Gibbons");

    word1.printInfo();
    word2.printInfo();
    word3.printInfo();
```

```
    //end successfully
    return 1;
}
```