```
/*
    Marshall Lindsay
    ECE 3220
    Lab8
    3/23/17
*/


#include <iostream>
#include <string>
#include <vector>

using namespace std;

class Message{
    protected:
        string msg;

    public:
        Message();
        Message(string);
        virtual ~Message();
        virtual void print();
};
/*
Default constructor;
Prompts the user for a new message and adds that message
to the msg string member
*/
Message::Message(){
    //cout<<"Constructor"<<endl;
    string userInput;
    cout<<"Please enter a message:"<<endl;
    getline(cin, userInput);

    //May need some error checking for special characters

    this -> msg = userInput;
}
/*
Parametric constructor;
Takes a string in as the message and adds that message to the
msg string member;
*/
Message::Message(string userInput){
    //cout<<"Constructor"<<endl;
    this -> msg = userInput;
}

/*Destructor.
Nothing in this class needs to be deleted;
*/
Message::~Message(){
    //cout<<"Destructor"<<endl;
}

void Message::print(){
    cout<<this ->msg<<endl;
}

//-----------------------------------------------------------------
//--------------Extended Message Class for Morse Code--------------
class morseCodeMessage: public Message{
    protected:
        string morseMsg;
    public:
        morseCodeMessage* next;
        morseCodeMessage();
        morseCodeMessage(string);
        ~morseCodeMessage();
        void translate(string);
        void print();
};
//Default constructor. Asks for a message and saves it as a string in
//the base classes msg member. Then translates that msg into morse code
//and saves the translation to the extended morseMsg member
morseCodeMessage::morseCodeMessage(){
    translate(this -> msg);
    this -> next = NULL;
}

//Parametric constructor. Takes a string as input. Calls the base
//constructor with the same string. Then translates the string
//to morse code and saves it to the extended morseMsg member.
morseCodeMessage::morseCodeMessage(string userInput) : Message(userInput){
    translate(this -> msg);
    this -> next = NULL;
}

morseCodeMessage::~morseCodeMessage(){
    //cout<<"Destructor"<<endl;
}

void morseCodeMessage::translate(string input){
    char test;
    for(int i = 0; i < input.length(); i++){
        input[i] = tolower(input[i]);
    }
    for(int i = 0; i < input.length(); i++){
        test = input[i];
        switch(test){
            case 'a':
                this->morseMsg.append(".- ");
                break;
            case 'b':
                this->morseMsg.append("-... ");
                break;
            case 'c':
                this->morseMsg.append("-.-. ");
                break;
            case 'd':
                this->morseMsg.append("-.. ");
                break;
            case 'e':
                this->morseMsg.append(". ");
                break;
            case 'f':
                this->morseMsg.append("..-. ");
                break;
            case 'g':
                this->morseMsg.append("--. ");
                break;
            case 'h':
                this->morseMsg.append(".... ");
                break;
            case 'i':
                this->morseMsg.append(".. ");
                break;
            case 'j':
                this->morseMsg.append(".--- ");
                break;
            case 'k':
                this->morseMsg.append("-.- ");
                break;
            case 'l':
                this->morseMsg.append(".-.. ");
                break;
            case 'm':
                this->morseMsg.append("-- ");
                break;
            case 'n':
                this->morseMsg.append("-. ");
                break;
            case 'o':
                this->morseMsg.append("--- ");
                break;
            case 'p':
                this->morseMsg.append(".--. ");
                break;
            case 'q':
                this->morseMsg.append("--.- ");
                break;
            case 'r':
                this->morseMsg.append(".-. ");
                break;
            case 's':
                this->morseMsg.append("... ");
                break;
            case 't':
                this->morseMsg.append("- ");
                break;
            case 'u':
                this->morseMsg.append("..- ");
                break;
            case 'v':
                this->morseMsg.append("...- ");
                break;
            case 'w':
                this->morseMsg.append(".-- ");
                break;
            case 'x':
                this->morseMsg.append("-..- ");
                break;
            case 'y':
                this->morseMsg.append("-.-- ");
                break;
            case 'z':
                this->morseMsg.append("--.. ");
                break;
            case '0':
                this->morseMsg.append("----- ");
                break;
            case '1':
                this->morseMsg.append(".---- ");
                break;
            case '2':
                this->morseMsg.append("..--- ");
                break;
            case '3':
                this->morseMsg.append("...-- ");
                break;
            case '4':
                this->morseMsg.append("....- ");
                break;
            case '5':
                this->morseMsg.append("..... ");
                break;
            case '6':
                this->morseMsg.append("-.... ");
                break;
            case '7':
                this->morseMsg.append("--... ");
                break;
            case '8':
                this->morseMsg.append("---.. ");
                break;
            case '9':
```

```cpp
                this->morseMsg.append("----. ");
                break;
            case ' ':
                this->morseMsg.append(" ");
                break;
            default:
                break;
        }//Switch

    }

}

void morseCodeMessage::print(){
    cout<< this->msg <<endl;
    cout<< this->morseMsg <<endl;
}

//------------------------------------------------------------------
//----------------------Generic Stack Class------------------------
class messageStack{
    private:
        morseCodeMessage* top;
    public:
        messageStack();
        ~messageStack();
        void push();
        void pop();
        void printStack();

};

//Default constructor. Sets the pointer to the top of the
//Stack to NULL.
messageStack::messageStack(){
    this -> top = NULL;
}

//Destructor. This will delete each object in the linked list.
messageStack::~messageStack(){
    morseCodeMessage* temp;
    while(top != NULL){
            temp = this -> top;
            this -> top = this -> top -> next;
            delete temp;
    }
}

//Creates a new message and sets it to the top of the stack.
void messageStack::push(){
    morseCodeMessage* newNode = new morseCodeMessage();
    newNode -> next = this -> top;
    this -> top = newNode;
}

//Cycles down the stack and prints each message in turn.
void messageStack::printStack(){
    morseCodeMessage* msgPtr = this -> top;
    while(msgPtr != NULL){
        msgPtr -> print();
        msgPtr = msgPtr -> next;
    }

}

//Deletes the top object from the stack. Sets the top of the
//stack to the next one down.
void messageStack::pop(){
    if(top == NULL){
        cout<<"\nThere is nothing on the stack!"<<endl;
        return;
    }
    morseCodeMessage* retObj = this -> top;
    this -> top = this -> top -> next;
    delete retObj;
}

void optionMenu();
void errorMessage();

int main(int argc, char* argv[]){
    messageStack stack;

    string userInput;

    while(1){
        optionMenu();
        getline(cin, userInput);
        if(userInput == "1"){
            stack.push();
        }else if(userInput == "2"){
            stack.pop();
        }else if(userInput == "3"){
            stack.printStack();
        }else if(userInput == "4"){
            return(0);
        }else{
            errorMessage();
        }
    }
}

void optionMenu(){
    cout<<"\nWhat would you like to do?"<<endl;
    cout<<"1) Push message to the stack\n"
        <<"2) Pop message off the stack\n"
        <<"3) Print all messages on the stack\n"
        <<"4) Quit"<<endl;
}

void errorMessage(){
    cout<<"Invalid input!"<<endl;
}
```