```
********************************************************************************
********************************************************************************
**                                                                            **
**                              pgm2.cc listing                               **
**                                                                            **
********************************************************************************
********************************************************************************

#include <iostream>
#undef NULL
const int NULL = 0;
const int BASE = 16;            //base unit of number we want to use
const char SENTINEL = '#';      //value of user_input that will end input
typedef int element;            //datatype of "elements" in LList
using namespace std;


/*
        Steven Liu
        CS215-J001
        Spring, 2011
        Program 2
*/


//*****************************global section*****************************


//global function prototypes
void displayMenu();
int charToInt(char input);
int charToASCII(char input);
char intToChar(int input);
char intToASCII(int input);
bool isBase(int base, char input);


//listnode class
        //each listnode consists of 2 sides:
        //1) one side, called "data" holds a single element
        //2) the other side, called "next" holds the address to the
        //next listnode
class listnode {
        public:
                element data;           //holds actual data
                listnode * next;        //holds address to next listnode
        };


//Linked List class
        //a valid linked list is defined as:
                //1) "head" points to the first listnode
                //2) followed by a series of listnodes
                //3) last listnode pointing to NULL
                //4) "tail" points to last listnode
        //when the list is empty (but also valid):
                //1) "head" points to NULL
                //2) "tail" is undefined
class LList {
        private:
                listnode * head;                //points to the first listnode
                listnode * tail;                //points to the last listnode
        public:
                //constructor/destructor:
                LList();        //constructor - auto called upon N.O. birth
                ~LList();       //destructor - auto called before N.O. death
                void PrintForward(int base, int total_ele);
                //methods:
                void Clean();
```



```
                void PrintBackward(int base);
                void InsertTail(element val);
                void InsertHead(element val);
                void Steal(LList & Victim);
                void Duplicate(LList & Source);
                int ReverveInPlace();   //extra credit
                //pgm2 specific methods
                void EnterNumber();
                void Add(LList & NewNum);
                void Multiply(LList & NewNum);
        };


//---------------------------End global section--------------------------------

//*****************************MAIN FUNCTION*****************************


int main(){
        char menu_choice;       //holds user menu choice
        LList CurrNum;          //holds current number, gets updated with answer
        LList NewNum;           //holds second number

        cout << "Hexadecimal calculator, Version 1.0" << endl
                << "(c) 2011, (Steven Liu)" << endl;

        //loop menu choice
        do {
                cout << endl << "Current hexadecimal number is: ";
                CurrNum.PrintBackward(BASE);
                cout << endl << "Command (h for help): ";
                cin >> menu_choice;
                switch (menu_choice) {
                        case 'a':
                        case 'A':
                                //addition
                                cout << endl << "Adding a new hexadecimal"
                                        << "number to the current hex. number."
                                        << endl;
                                NewNum.EnterNumber();
                                cout << endl << "New hexadecimal number is: ";
                                NewNum.PrintBackward(BASE);
                                CurrNum.Add(NewNum);
                                cout << endl << "Adding completed." << endl;
                                break;
                        case 'e':
                        case 'E':
                                //enter new number into CurrNum
                                CurrNum.EnterNumber();
                                cout << endl << "Entering completed." << endl;
                                break;
                        case 'h':
                        case 'H':
                                //help
                                displayMenu();
                                break;
                        case 'm':
                        case 'M':
                                //multiplication
                                cout << endl << "Multiplying a new hexadecimal"
                                        << "number to the current hex. number."
                                        << endl;
                                NewNum.EnterNumber();
                                cout << endl << "New hexadecimal number is: ";
                                NewNum.PrintBackward(BASE);
                                CurrNum.Multiply(NewNum);
                                cout << endl << "Multiplying completed."
                                        << endl;
                                break;
```

```
                        case 'q':
                        case 'Q':
                                //quit
                                cout << endl << "Finishing Hexadecimal"
                                        << "Calculator, version 1.0" << endl;
                                break;
                        default:
                                cout << endl << "Not a valid menu choice,"
                                        << " please try again." << endl;
                                break;
                        }
                } while ((menu_choice != 'q') && (menu_choice != 'Q'));
        }


//---------------------------END MAIN FUNCTION-------------------------------

//***************************global functions*****************************


void displayMenu() {
        //displays options menu

        cout << endl << "Valid commands are:" << endl
                << "   e   enter     enter the current hexadecimal"
                << " number from the keyboard" << endl
                << "   a   add       add a new hexadecimal number to the"
                << " current hex. number" << endl
                << "   m   multiply  multiply a new hexadecimal number"
                << " to the current hex. number" << endl
                << "   h   help      show this help menu" << endl
                << "   q   quit      quit the program" << endl;
        }


int charToInt(char input) {
        //pre: "input" must be valid and hold a character
        //post: an integer "output", containing the decimal representation
                //of "input" will be returned

        int output;             //holds the integer representation of "input"

        if ((input >= 'a') && (input <= 'z'))
                output = input - 87;
        else if ((input >= 'A') && (input <= 'Z'))
                output = input - 55;
        else if ((input >= '0') && (input <= '9'))
                output = input - 48;
        else //is a char other than a-z, A-Z, 0-9
                output = -1;

        return output;
        }


int charToASCII(char input) {
        //pre: "input" must be valid and hold an ASCII
        //post: an integer "output", containing the decimal representation
                //of "input" will be returned

        int output;
        output = input;

        return output;
        }


char intToChar(int input) {
        //pre: "input" must be valid and hold an integer
```



```
        //post: a character "output", containing the character representation
                //of "input" will be returned

        char output;

        if ((input >= 0) && (input <= 9))
                output = input + 48;
        else if ((input >= 10) && (input <= 35))
                output = input + 55;
        else //is a char other than a-z, A-Z, 0-9
                output = '?';

        return output;
        }


char intToASCII(int input) {
        //pre: "input" must be valid and hold an integer
        //post: a character "output", containing the ASCII representation
                //of "input" will be returned

        char output;
        output = input;

        return output;
        }


bool isBase(int base, char input) {
        //pre: "base" and "input" must be valid
        //post: returns true if "input" is a value between 0 and "base"
                //returns false otherwise

        char charCode;
        charCode = charToInt(input);

        return ((charCode >= 0) && (charCode < base));
        }


//---------------------------End global functions-----------------------------

//***********************LList constructor/destructor***********************


//constructor
LList::LList() {
        //pre: none
        //post: the N.O. LList is empty

        head = NULL;
        }


//destructor
LList::~LList() {
        //pre: the N.O. LList is valid
        //post: the N.O. LList is empty

        Clean();
        }


//-------------------End LList constructor/destructor-----------------------

//*****************************LList methods*********************************


//cleans the LList of all nodes
```

```
void LList::Clean() {
        //pre: N.O. is valid
        //post: N.O. is now empty and all of its former listnodes have
                //had their memory returned to the system memory pool

        listnode * temp;                //points listnode to be deleted

        //we point "head" at the next listnode, maintaining a valid LList
        //while "temp" points to the listnode we want to delete
        while (head != NULL) {
                temp = head;
                head = head->next;
                delete temp;
                }
        }


//prints out the entire LList forwards
void LList::PrintForward(int base, int total_ele) {
        //pre: N.O. is valid
        //post: N.O. is unchanged, and the element it contains
                //have been displayed forwards

        listnode * temp;        //LCV - pointer to traverse list
        int less_head;          //holds # total elements's % 3
        int position;           //accum - maintains element position

        temp = head;
        less_head = total_ele % 3;
        position = 1;

        if ((base >= 0) && (base <= 36))
                if (temp == NULL)
                        cout << 0;
                else //has content
                        while (temp != NULL) {
                                cout << intToChar(temp->data);
                                if ( ((position - less_head) % 3 == 0)
                                        && (temp->next != NULL) )
                                        cout << ",";
                                else
                                        ;
                                temp = temp->next;      //pointer increment
                                position++;
                                }
        else //is ASCII
                if (temp == NULL)
                        cout << "[None!]";
                else //has content
                        while (temp != NULL) {
                                cout << "'" << intToASCII(temp->data) << "' ";
                                temp = temp->next;      //pointer increment
                                }

        cout << endl;
        }


//prints out the entire LList backwards
void LList::PrintBackward(int base) {
        //pre: N.O. is valid
        //post: N.O. is unchanged, and the element it contains
                //have been displayed backwards

        PrintForward(base, ReverveInPlace());
        ReverveInPlace();
        }
```



```
//inserts one listnode containing element val, at the END of list
void LList::InsertTail(element val) {
        //pre: N.O. is valid, and element "val" is valid
        //post: N.O. is unchanged, except it now has an addition
                //listnode at its tail-end containing element val

        listnode * temp;                //points to new listnode

        temp = new listnode;
        temp->data = val;
        temp->next = NULL;
        if (head == NULL) //empty list
                head = temp;
        else //not empty list
                tail->next = temp;
        tail = temp;
        }


//inserts one listnode containing element val, at the FRONT of list
void LList::InsertHead(element val) {
        //pre: N.O. is valid, and element "val" is valid
        //post: N.O. is unchanged, except it now has an addition
                //listnode at its head-end containing element val

        listnode * temp;                //points to new listnode

        temp = new listnode;
        temp->data = val;
        temp->next = head;
        if (head == NULL) //empty list
                tail = temp;
        else //not empty list
                ;
        head = temp;
        }


//takes over "Victim"'s listnodes after throwing away N.O.'s listnodes
void LList::Steal(LList & Victim) {
        //pre: N.O. is valid, the LList Victim is valid
        //post: N.O. has returned all of its memory to system pool (heap)
                //and now contains the listnodes originally on the
                //Victim LList.  The Victim LList is empty

        Clean();        //N.O.'s Clean() method - removes all self listnodes

        //if the visiting/local object's class names are the same as N.O.'s,
                //all co-members of the visiting/local object are visible
                //to N.O. (including private co-members)
        head = Victim.head;
        tail = Victim.tail;
        Victim.head = NULL;
        }


//makes N.O. an exact copy of "Source"
void LList::Duplicate(LList & Source) {
        //pre: N.O. is valid, the LList Source is valid
        //post: LList Source is valid.  N.O. will be an exact
                //listnode for listnode copy of Source

        Clean();        //removes any existing listnodes in linked list

        listnode * temp;
        temp = Source.head;
        while (temp != NULL) {
                InsertTail(temp->data);         //N.O.'s InsertTail() method
                temp = temp->next;
```

```
                }
        }


//reverses the listnodes in the N.O. LList − cannot use extra memory space
int LList::ReverveInPlace() {   //extra credit
        //pre: the N.O. is valid
        //post: the N.O. is unchanged, except elements in its listnodes
                //are now in reverse order
        int total_ele;

        if (head == NULL)
                total_ele = 0; //no elements
        else if (head->next == NULL)
                total_ele = 1; //one element
        else {
                //since we're inside of the else statement,
                        //there MUST be at least 2 listnodes in the LList
                total_ele = 1;

                listnode * prev;                //points to previous listnode
                listnode * curr;                //points to current listnode
                listnode * succ;                //points to succeeding listnode

                prev = head;
                curr = head->next;
                succ = curr->next;

                //since there are at least 2 listnodes, we have to reverse
                        //listnodes (loop body) at least once − dowhile loop
                //we're done when:
                        //prev == tail OR curr == NULL, only need to pick one
                        //becase we increment both prev and curr every loop
                do {
                        curr->next = prev;      //reverse listnode
                        total_ele++;

                        //pointer increments:
                        prev = curr;
                        curr = succ;
                        if (succ != NULL)
                                succ = succ->next;
                        else
                                ;
                        } while (prev != tail);
                //by end of the above loop we know:
                        //directions of all listnodes have been reversed
                        //but the two ends of the listnodes aren't clear
                //however, we know that:
                        //1) head is currently pointing to new tail
                        //2) tail is currently pointing to new head
                        //3) prev is also pointing to new head

                tail = head;
                tail->next = NULL;
                head = prev;
                }

                return total_ele;
        }


//reads in data
void LList::EnterNumber(){
        //pre: N.O. is valid, Rejected is valid
        //post: all user_input that are hexadecimal are stored
                //in the N.O., while any user_input that aren't
                //hexadecimal are stored in Rejected
```



```
        char user_input;        //individual user keyboard input
        LList Rejected;         //holds rejected keyboard input

        Clean();        //removes all existing listnodes

        cout << endl << "Enter a hexadecimal number, followed by "
                << SENTINEL << ": ";

        cin >> user_input;
        while (user_input != SENTINEL) {
                if ( isBase(BASE, user_input) )
                        InsertHead( charToInt(user_input) );
                else //not hex
                        Rejected.InsertHead( charToASCII(user_input) );
                cin >> user_input;
                }
        //by end of the above loop, we know:
                //user has entered the SENTINEL value
                //N.O. contains all the valid BASE user_inputs,
                //while Rejected contains any user_inputs that aren't valid

        cout << endl << "Rejected inputs: ";
        Rejected.PrintBackward(-1);
        }


//performs addition on N.O. and NewNum
void LList::Add(LList & NewNum) {
        //pre: NO and NewNum are valid
        //post: NewNum remains unchanged, while NO now contains the sum of
                //the contents inside NO and NewNum

        LList TotalSum;         //holds total sum
        listnode * temp1;       //holds pointer position of N.O.
        listnode * temp2;       //holds pointer position of NewNum
        int sum;                //holds sum to current column
        int carry;              //holds carry overs
        int answer;             //holds answer to current column

        temp1 = head;
        temp2 = NewNum.head;

        carry = 0;

        //phase 1, add numbers when both LList has numbers
        while ((temp1 != NULL) && (temp2 != NULL)){
                sum = carry + temp1->data + temp2->data;

                carry = sum / BASE;
                answer = sum % BASE;

                TotalSum.InsertTail(answer);
                temp1 = temp1->next;
                temp2 = temp2->next;
                }
        //by the end of the phase 1 loop we know:
                //one or both LLists are out of numbers, but we need to continue
                //further to make sure we take care of any LList that may still
                //have numbers left

        //phase 2a, add numbers when NO still has numbers left
        while (temp1 != NULL){
                sum = carry + temp1->data;

                carry = sum / BASE;
                answer = sum % BASE;

                TotalSum.InsertTail(answer);
```

```
                temp1 = temp1->next;
                }
        //phase 2b, add numbers when NewNum still has numbers left
        while (temp2 != NULL){
                sum = carry + temp2->data;

                carry = sum / BASE;
                answer = sum % BASE;

                TotalSum.InsertTail(answer);

                temp2 = temp2->next;
                }
        //by the end of the phase 2 loops we know:
                //both LLists are out of numbers, but we still
                //could have a carry > 0 that needs to be added

        //phase 3, tag on carry if the last addition was bigger than BASE
        if (carry != 0)
                TotalSum.InsertTail(carry);
        else
                ;
        //by the end of phase 3 know:
                //both LLists are out of numbers,
                //all numbers have been added together
                //any carry > 0 have been accounted for

        Steal(TotalSum);
        }


//performs multiplication on N.O. and NewNum
void LList::Multiply(LList & NewNum) {
        //pre: NO and NewNum are valid
        //post: NewNum remains unchanged, while NO now contains the product
                //of the contents inside NO and NewNum

        LList TotalProduct;     //holds total product
        LList ColProduct;       //holds col product

        listnode * temp;        //holds pointer position of N.O.
        temp = head;

        while (temp != NULL){
                if (temp->data == 0) //col = 0
                        ColProduct.Clean();
                else { //col != 0
                        ColProduct.Duplicate(NewNum);
                        for (int i = 1; i < temp->data; i++)
                                ColProduct.Add(NewNum);
                        }
                //by the end of the above if/else we know:
                        //ColProduct contains
                        //the product of NO's current col and NewNum

                TotalProduct.Add(ColProduct);

                NewNum.InsertHead(0);
                temp = temp->next;
                }

        Steal(TotalProduct);
        }


//--------------------------End LList methods-------------------------------
```

```
********************************************************************************
********************************************************************************
**                                                                            **
**                            pgm2.cc compilation                             **
**                                                                            **
********************************************************************************
********************************************************************************

c++ compilation succeeded
```



```
********************************************************************************
********************************************************************************
**                                                                            **
**                  pgm2.cc execution - required testcase #1                  **
**                                                                            **
********************************************************************************
********************************************************************************

Hexadecimal calculator, Version 1.0
(c) 2011, (Steven Liu)

Current hexadecimal number is: 0

Command (h for help): h

Valid commands are:
   e   enter     enter the current hexadecimal number from the keyboard
   a   add       add a new hexadecimal number to the current hex. number
   m   multiply  multiply a new hexadecimal number to the current hex. number
   h   help      show this help menu
   q   quit      quit the program

Current hexadecimal number is: 0

Command (h for help): e

Enter a hexadecimal number, followed by #: 4b7c
#

Rejected inputs: [None!]

Entering completed.

Current hexadecimal number is: 4,B7C

Command (h for help): a

Adding a new hexadecimalnumber to the current hex. number.

Enter a hexadecimal number, followed by #: 98FA7#

Rejected inputs: [None!]

New hexadecimal number is: 98,FA7

Adding completed.

Current hexadecimal number is: 9D,B23

Command (h for help): q

Finishing HexadecimalCalculator, version 1.0
```

```
**********************************************************************************
**********************************************************************************
**                                                                              **
**                  pgm2.cc execution - required testcase #2                    **
**                                                                              **
**********************************************************************************
**********************************************************************************

Hexadecimal calculator, Version 1.0
(c) 2011, (Steven Liu)

Current hexadecimal number is: 0

Command (h for help): e

Enter a hexadecimal number, followed by #: 9BB1D#

Rejected inputs: [None!]

Entering completed.

Current hexadecimal number is: 9B,B1D

Command (h for help): m

Multiplying a new hexadecimalnumber to the current hex. number.

Enter a hexadecimal number, followed by #: C2ba#

Rejected inputs: [None!]

New hexadecimal number is: C,2BA

Multiplying completed.

Current hexadecimal number is: 766,DDE,D12

Command (h for help): q

Finishing HexadecimalCalculator, version 1.0
```



```
**********************************************************************************
**********************************************************************************
**                                                                              **
**                  pgm2.cc execution - required testcase #3                    **
**                                                                              **
**********************************************************************************
**********************************************************************************

Hexadecimal calculator, Version 1.0
(c) 2011, (Steven Liu)

Current hexadecimal number is: 0

Command (h for help): e

Enter a hexadecimal number, followed by #: 123456789ABCDEF#

Rejected inputs: [None!]

Entering completed.

Current hexadecimal number is: 123,456,789,ABC,DEF

Command (h for help): a

Adding a new hexadecimalnumber to the current hex. number.

Enter a hexadecimal number, followed by #: 0#

Rejected inputs: [None!]

New hexadecimal number is: 0

Adding completed.

Current hexadecimal number is: 123,456,789,ABC,DEF

Command (h for help): q

Finishing HexadecimalCalculator, version 1.0
```

```
**********************************************************************************
**********************************************************************************
**                                                                              **
**                   pgm2.cc execution - required testcase #4                   **
**                                                                              **
**********************************************************************************
**********************************************************************************

Hexadecimal calculator, Version 1.0
(c) 2011, (Steven Liu)

Current hexadecimal number is: 0

Command (h for help): e

Enter a hexadecimal number, followed by #: FFFFFFFFFFFFFFF#

Rejected inputs: [None!]

Entering completed.

Current hexadecimal number is: F,FFF,FFF,FFF,FFF,FFF

Command (h for help): A

Adding a new hexadecimalnumber to the current hex. number.

Enter a hexadecimal number, followed by #: 1#

Rejected inputs: [None!]

New hexadecimal number is: 1

Adding completed.

Current hexadecimal number is: 10,000,000,000,000,000

Command (h for help): q

Finishing HexadecimalCalculator, version 1.0
```



```
**********************************************************************************
**********************************************************************************
**                                                                              **
**                   pgm2.cc execution - required testcase #5                   **
**                                                                              **
**********************************************************************************
**********************************************************************************

Hexadecimal calculator, Version 1.0
(c) 2011, (Steven Liu)

Current hexadecimal number is: 0

Command (h for help): e

Enter a hexadecimal number, followed by #: 123456789abcdef#

Rejected inputs: [None!]

Entering completed.

Current hexadecimal number is: 123,456,789,ABC,DEF

Command (h for help): m

Multiplying a new hexadecimalnumber to the current hex. number.

Enter a hexadecimal number, followed by #: 1#

Rejected inputs: [None!]

New hexadecimal number is: 1

Multiplying completed.

Current hexadecimal number is: 123,456,789,ABC,DEF

Command (h for help): q

Finishing HexadecimalCalculator, version 1.0
```

```
********************************************************************************
********************************************************************************
**                                                                            **
**                  pgm2.cc execution - required testcase #6                  **
**                                                                            **
********************************************************************************
********************************************************************************

Hexadecimal calculator, Version 1.0
(c) 2011, (Steven Liu)

Current hexadecimal number is: 0

Command (h for help): e

Enter a hexadecimal number, followed by #: 123456789AbCdEf#

Rejected inputs: [None!]

Entering completed.

Current hexadecimal number is: 123,456,789,ABC,DEF

Command (h for help): m

Multiplying a new hexadecimalnumber to the current hex. number.

Enter a hexadecimal number, followed by #: 0#

Rejected inputs: [None!]

New hexadecimal number is: 0

Multiplying completed.

Current hexadecimal number is: 000,000,000,000,000

Command (h for help): q

Finishing HexadecimalCalculator, version 1.0
```



```
********************************************************************************
********************************************************************************
**                                                                            **
**                  pgm2.cc execution - required testcase #7                  **
**                                                                            **
********************************************************************************
********************************************************************************

Hexadecimal calculator, Version 1.0
(c) 2011, (Steven Liu)

Current hexadecimal number is: 0

Command (h for help): e

Enter a hexadecimal number, followed by #:  1F0BF92CEC337FD1E319552ABFC4525BA419
28CCF09645F4B52D5BF30#

Rejected inputs: [None!]

Entering completed.

Current hexadecimal number is: 1F0,BF9,2CE,C33,7FD,1E3,195,52A,BFC,452,5BA,419,2
8C,CF0,964,5F4,B52,D5B,F30

Command (h for help): a

Adding a new hexadecimalnumber to the current hex. number.

Enter a hexadecimal number, followed by #: 39E4784EBC76AF17CB5130F15BB9845BF5A2F
86AB1689FCE1A0E317C4#

Rejected inputs: [None!]

New hexadecimal number is: 39E,478,4EB,C76,AF1,7CB,513,0F1,5BB,984,5BF,5A2,F86,A
B1,689,FCE,1A0,E31,7C4

Adding completed.

Current hexadecimal number is: 58F,071,7BA,8AA,2EE,9AE,6A8,61C,1B7,DD6,B79,9BC,2
13,7A1,FEE,5C2,CF3,B8D,6F4

Command (h for help): q

Finishing HexadecimalCalculator, version 1.0
```

```
**********************************************************************************
**********************************************************************************
**                                                                              **
**                   pgm2.cc execution - required testcase #8                   **
**                                                                              **
**********************************************************************************
**********************************************************************************

Hexadecimal calculator, Version 1.0
(c) 2011, (Steven Liu)

Current hexadecimal number is: 0

Command (h for help): e

Enter a hexadecimal number, followed by #: 1111111111111111111111111111111111111111
1111111111111111#

Rejected inputs: [None!]

Entering completed.

Current hexadecimal number is: 11,111,111,111,111,111,111,111,111,111,111,111,11
1,111,111,111,111,111

Command (h for help): m

Multiplying a new hexadecimalnumber to the current hex. number.

Enter a hexadecimal number, followed by #: FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFF#

Rejected inputs: [None!]

New hexadecimal number is: FF,FFF,FFF,FFF,FFF,FFF,FFF,FFF,FFF,FFF,FFF,FFF,FFF,FF
F,FFF,FFF,FFF,FFF

Multiplying completed.

Current hexadecimal number is: 1,111,111,111,111,111,111,111,111,111,111,111,111
,111,111,111,111,0EE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE,EEE
,EEE,EEE,EEF

Command (h for help): q

Finishing HexadecimalCalculator, version 1.0
```



```
**********************************************************************************
**********************************************************************************
**                                                                              **
**                   pgm2.cc execution - required testcase #9                   **
**                                                                              **
**********************************************************************************
**********************************************************************************

Hexadecimal calculator, Version 1.0
(c) 2011, (Steven Liu)

Current hexadecimal number is: 0

Command (h for help): e

Enter a hexadecimal number, followed by #: 17maybe247-+.7466#

Rejected inputs: 'm' 'y' '-' '+' '.'

Entering completed.

Current hexadecimal number is: 17A,BE2,477,466

Command (h for help): q

Finishing HexadecimalCalculator, version 1.0
```

```
********************************************************************************
********************************************************************************
**                                                                            **
**                  pgm2.cc execution − required testcase #10                 **
**                                                                            **
********************************************************************************
********************************************************************************

Hexadecimal calculator, Version 1.0
(c) 2011, (Steven Liu)

Current hexadecimal number is: 0

Command (h for help): e

Enter a hexadecimal number, followed by #: 17maybe247−+.7466@#

Rejected inputs: 'm' 'y' '−' '+' '.' '@'

Entering completed.

Current hexadecimal number is: 17A,BE2,477,466

Command (h for help): q

Finishing HexadecimalCalculator, version 1.0
```



```
********************************************************************************
********************************************************************************
**                                                                            **
**            pgm2.cc execution − proposed normal testcase 1 [#1]             **
**                                                                            **
********************************************************************************
********************************************************************************

Hexadecimal calculator, Version 1.0
(c) 2011, (Steven Liu)

Current hexadecimal number is: 0

Command (h for help): e

Enter a hexadecimal number, followed by #: 1#

Rejected inputs: [None!]

Entering completed.

Current hexadecimal number is: 1

Command (h for help): a

Adding a new hexadecimalnumber to the current hex. number.

Enter a hexadecimal number, followed by #: 1#

Rejected inputs: [None!]

New hexadecimal number is: 1

Adding completed.

Current hexadecimal number is: 2

Command (h for help): a

Adding a new hexadecimalnumber to the current hex. number.

Enter a hexadecimal number, followed by #: 10#

Rejected inputs: [None!]

New hexadecimal number is: 10

Adding completed.

Current hexadecimal number is: 12

Command (h for help): a

Adding a new hexadecimalnumber to the current hex. number.

Enter a hexadecimal number, followed by #: A#

Rejected inputs: [None!]

New hexadecimal number is: A

Adding completed.

Current hexadecimal number is: 1C

Command (h for help): a

Adding a new hexadecimalnumber to the current hex. number.
```

```
Enter a hexadecimal number, followed by #: D2#
Rejected inputs: [None!]
New hexadecimal number is: D2
Adding completed.
Current hexadecimal number is: EE
Command (h for help): e
Enter a hexadecimal number, followed by #: e#
Rejected inputs: [None!]
Entering completed.
Current hexadecimal number is: E
Command (h for help): m
Multiplying a new hexadecimalnumber to the current hex. number.
Enter a hexadecimal number, followed by #: 9#
Rejected inputs: [None!]
New hexadecimal number is: 9
Multiplying completed.
Current hexadecimal number is: 7E
Command (h for help): a
Adding a new hexadecimalnumber to the current hex. number.
Enter a hexadecimal number, followed by #: 9#
Rejected inputs: [None!]
New hexadecimal number is: 9
Adding completed.
Current hexadecimal number is: 87
Command (h for help): m
Multiplying a new hexadecimalnumber to the current hex. number.
Enter a hexadecimal number, followed by #: 7a#
Rejected inputs: [None!]
New hexadecimal number is: 7A
Multiplying completed.
Current hexadecimal number is: 4,056
Command (h for help): e
Enter a hexadecimal number, followed by #: 100#
Rejected inputs: [None!]
Entering completed.
```



```
Current hexadecimal number is: 100
Command (h for help): q
Finishing HexadecimalCalculator, version 1.0
```

```
**************************************************************************
**************************************************************************
**                                                                      **
**           pgm2.cc execution - proposed boundary testcase 1 [#1]      **
**                                                                      **
**************************************************************************
**************************************************************************

Hexadecimal calculator, Version 1.0
(c) 2011, (Steven Liu)

Current hexadecimal number is: 0

Command (h for help): e

Enter a hexadecimal number, followed by #: 0#

Rejected inputs: [None!]

Entering completed.

Current hexadecimal number is: 0

Command (h for help): a

Adding a new hexadecimalnumber to the current hex. number.

Enter a hexadecimal number, followed by #: 0#

Rejected inputs: [None!]

New hexadecimal number is: 0

Adding completed.

Current hexadecimal number is: 0

Command (h for help): m

Multiplying a new hexadecimalnumber to the current hex. number.

Enter a hexadecimal number, followed by #: FF#

Rejected inputs: [None!]

New hexadecimal number is: FF

Multiplying completed.

Current hexadecimal number is: 0

Command (h for help): e

Enter a hexadecimal number, followed by #: 9#

Rejected inputs: [None!]

Entering completed.

Current hexadecimal number is: 9

Command (h for help): a

Adding a new hexadecimalnumber to the current hex. number.

Enter a hexadecimal number, followed by #: 9#

Rejected inputs: [None!]
```



```
New hexadecimal number is: 9

Adding completed.

Current hexadecimal number is: 12

Command (h for help): a

Adding a new hexadecimalnumber to the current hex. number.

Enter a hexadecimal number, followed by #: ED#

Rejected inputs: [None!]

New hexadecimal number is: ED

Adding completed.

Current hexadecimal number is: FF

Command (h for help): e

Enter a hexadecimal number, followed by #: f#

Rejected inputs: [None!]

Entering completed.

Current hexadecimal number is: F

Command (h for help): A

Adding a new hexadecimalnumber to the current hex. number.

Enter a hexadecimal number, followed by #: F#

Rejected inputs: [None!]

New hexadecimal number is: F

Adding completed.

Current hexadecimal number is: 1E

Command (h for help): a

Adding a new hexadecimalnumber to the current hex. number.

Enter a hexadecimal number, followed by #: F0#

Rejected inputs: [None!]

New hexadecimal number is: F0

Adding completed.

Current hexadecimal number is: 10E

Command (h for help): M

Multiplying a new hexadecimalnumber to the current hex. number.

Enter a hexadecimal number, followed by #: 0#

Rejected inputs: [None!]

New hexadecimal number is: 0

Multiplying completed.
```

```
Current hexadecimal number is: 000

Command (h for help): q

Finishing HexadecimalCalculator, version 1.0
```



```
********************************************************************************
********************************************************************************
**                                                                            **
**           pgm2.cc execution − proposed exception testcase 1 [#1]           **
**                                                                            **
********************************************************************************
********************************************************************************

Hexadecimal calculator, Version 1.0
(c) 2011, (Steven Liu)

Current hexadecimal number is: 0

Command (h for help): e

Enter a hexadecimal number, followed by #: xyz#

Rejected inputs: 'x' 'y' 'z'

Entering completed.

Current hexadecimal number is: 0

Command (h for help): e

Enter a hexadecimal number, followed by #: 8jja9b#

Rejected inputs: 'j' 'j'

Entering completed.

Current hexadecimal number is: 8,A9B

Command (h for help): a

Adding a new hexadecimalnumber to the current hex. number.

Enter a hexadecimal number, followed by #: zzz#

Rejected inputs: 'z' 'z' 'z'

New hexadecimal number is: 0

Adding completed.

Current hexadecimal number is: 8,A9B

Command (h for help): m

Multiplying a new hexadecimalnumber to the current hex. number.

Enter a hexadecimal number, followed by #: −13#

Rejected inputs: '−'

New hexadecimal number is: 13

Multiplying completed.

Current hexadecimal number is: A4,981

Command (h for help): e

Enter a hexadecimal number, followed by #: .1#

Rejected inputs: '.'

Entering completed.
```

```
Current hexadecimal number is: 1
Command (h for help): Z
Not a valid menu choice, please try again.
Current hexadecimal number is: 1
Command (h for help): e
Enter a hexadecimal number, followed by #: mkd83oa#
Rejected inputs: 'm' 'k' 'o'
Entering completed.
Current hexadecimal number is: D,83A
Command (h for help): 9
Not a valid menu choice, please try again.
Current hexadecimal number is: D,83A
Command (h for help): A
Adding a new hexadecimalnumber to the current hex. number.
Enter a hexadecimal number, followed by #: 1.008#
Rejected inputs: '.'
New hexadecimal number is: 1,008
Adding completed.
Current hexadecimal number is: E,842
Command (h for help): m
Multiplying a new hexadecimalnumber to the current hex. number.
Enter a hexadecimal number, followed by #: %$*jam0#
Rejected inputs: '%' '$' '*' 'j' 'm'
New hexadecimal number is: A0
Multiplying completed.
Current hexadecimal number is: 912,940
Command (h for help): q
Finishing HexadecimalCalculator, version 1.0
```