**University of Michigan – Dearborn**

**CIS 200 – Computer Science 2**

**Lab 08**

Nahrin Sharna

[nsharna@umich.edu](mailto:nsharna@umich.edu)

03/18/2019

# Source Code:

/\*

Author: Nahrin Sharna

Creation Date: 03/16/2019

Modification: 03/18/2019

Purpose: Create a doubly linked list, print the full sorted list and also sorted list with even numbers only

\*/

#include<iostream>

#include<fstream>

#include<string>

using namespace std;

int c = 0;

struct node

{

node \*next, \*prev;

node \*evenNxt, \*evenPrev;

int data;

}\*head = NULL, \*tail = NULL, \*p = NULL, \*r = NULL, \*np = NULL;

//creating lists

void create(int x)

{

np = new node;

np->data = x;

np->next = NULL;

np->prev = NULL;

np->evenNxt = NULL;

np->evenPrev = NULL;

if (c == 0)

{

tail = np;

head = np;

p = head;

p->next = NULL;

p->prev = NULL;

p->evenNxt = NULL;

p->evenPrev = NULL;

c++;

}

else

{

p = head;

r = p;

if (np->data < p->data)

{

np->next = p;

p->prev = np;

np->prev = NULL;

if ((np->data) % 2 == 1) {

if ((p->data) % 2 == 1) {

np->evenNxt = p;

p->evenPrev = np;

np->evenPrev = NULL;

}

}

head = np;

p = head;

do

{

p = p->next;

}

while (p->next != NULL);

tail = p;

}

else if (np->data > p->data)

{

while (p != NULL && np->data > p->data)

{

r = p;

p = p->next;

if (p == NULL)

{

r->next = np;

np->prev = r;

np->next = NULL;

if ((np->data) % 2 == 1) {

if ((r->data) % 2 == 1) {

np->evenPrev = r;

r->evenNxt = np;

r->evenPrev = NULL;

}

}

tail = np;

break;

}

else if (np->data < p->data)

{

r->next = np;

np->prev = r;

np->next = p;

p->prev = np;

if (p->next != NULL)

{

do

{

p = p->next;

} while (p->next != NULL);

}

tail = p;

break;

}

}

}

}

}

//printing full list in descending

void traverse\_tail()

{

node \*t = tail;

while (t != NULL)

{

cout << t->data << "\t";

t = t->prev;

}

cout << endl;

}

//printing full list in ascending order

void traverse\_head()

{

node \*t = head;

while (t != NULL)

{

cout << t->data << "\t";

t = t->next;

}

cout << endl;

}

//printing even list in ascending order

void traverseEven\_head() {

node \*t1 = head;

while (t1 != NULL) {

if ((t1->data) % 2 == 1) {

cout << t1->data << " ";

t1 = t1->evenNxt;

}

}

}

//printing even list in descending order

void traverseEven\_tail() {

node \*t2 = tail;

while (t2 != NULL) {

if((t2->data) % 2 == 1) {

cout << t2->data << " ";

t2 = t2->evenPrev;

}

}

}

//check if the input value of integer already exists in the list or not

bool valDoesExist(int x) {

node \*trv = head;

while (trv != NULL) {

if (trv->data == x) {

return true;

}

trv = trv->next;

}

return false;

}

// add new numbers in the existing list

void addNumber(int x) {

node \*add = head;

char choice;

while (add->next != NULL) {

if (x > 0) {

np = new node;

np->data = x;

np->next = NULL;

np->prev = NULL;

if (c == 0)

{

tail = np;

head = np;

p = head;

p->next = NULL;

p->prev = NULL;

c++;

}

else

{

p = head;

r = p;

if (np->data < p->data)

{

np->next = p;

p->prev = np;

np->prev = NULL;

head = np;

p = head;

do

{

p = p->next;

}

while (p->next != NULL);

tail = p;

}

else if (np->data > p->data)

{

while (p != NULL && np->data > p->data)

{

r = p;

p = p->next;

if (p == NULL)

{

r->next = np;

np->prev = r;

np->next = NULL;

tail = np;

break;

}

else if (np->data < p->data)

{

r->next = np;

np->prev = r;

np->next = p;

p->prev = np;

if (p->next != NULL)

{

do

{

p = p->next;

}

while (p->next != NULL);

}

tail = p;

}

}

}

}

cout << "Do you want to print the new list in ascending or descending: ";

cin >> choice;

if (choice == 'A') {

cout << "The new list: " << endl;

traverse\_head();

}

else if (choice == 'D') {

cout << "The new list: " << endl;

traverse\_tail();

}

else {

cout << "Invalid option";

}

break;

}

}

}

// delete an integer from the existing list

void deleteInt(int x) {

node \*dlt; node \*curr;

{

if (head->data == x) {

dlt = head;

head = head->next;

delete dlt;

}

else {

curr = head;

while (curr->next->data != x) {

curr = curr->next;

}

dlt = curr->next;

curr->next = curr->next->next;

delete dlt;

}

cout << "The new list is: " << endl;

while (head) {

cout << head->data << " ";

head = head->next;

}

}

}

int main() {

ifstream ins;

ofstream outs;

outs.open("errorlog.txt");

int numbers;

ins.open("integer.dat");

char choice;

int num1;

cout << "------------------------------------------------------------------------------------------" << endl;

cout << " WELCOME " << endl;

cout << "------------------------------------------------------------------------------------------" << endl;

if (!ins.is\_open()) {

cout << "File doesn't exists." << endl;

outs << "File doesn't exists." << endl;

}

else if (ins.peek() == EOF) {

cout << "File is empty." << endl;

outs << "File is empty." << endl;

}

else {

while (!ins.eof()) {

ins >> numbers;

if (numbers < 1) {

cout << "Numbers can't be less than 1" << endl;

outs << "Numbers can't be less than 1" << endl;

}

else {

create(numbers);

}

}

//printing full list and even list in sorted orders

cout << "Do you want to print the list in Ascending or Descending order(A or D): ";

cin >> choice;

if (toupper(choice) == 'A') {

traverse\_head();

cout << "Even List in ascending: ";

traverseEven\_head();

}

else if (toupper(choice) == 'D') {

traverse\_tail();

cout << "Even List in descending: ";

traverseEven\_tail();

}

else

{

cout << "Invalid option" << endl;

outs << "Invalid option" << endl;

}

cout << endl;

//adding new element to the sorted list

cout << "Do you want to add numbers: ";

cin >> choice;

if (choice == 'Y') {

cout << "Enter the number: ";

cin >> num1;

bool s = valDoesExist(num1);

if (num1 <= 0) {

cout << "Numbers can't be zero or negatives." << endl;

outs << "Numbers can't be zero or negatives." << endl;

}

else if (s == true) {

cout << "Number already exists in the list." << endl;

outs << "Number already exists in the list." << endl;

}

else {

addNumber(num1);

}

}

else if (choice == 'N') {

cout << "No new number is added." << endl;

}

else {

cout << "Invalid option chosen." << endl;

outs << "Invalid option chosen." << endl;

}

//deleting element

cout << "Do you want to delete any integer(Y or N): " << endl;

cin >> choice;

if (choice == 'Y') {

cout << "Enter the integer to delete: ";

cin >> num1;

bool s1 = valDoesExist(num1);

if (s1 == false) {

cout << "Integer does not exist in the list." << endl;

outs << "Integer does not exist in the list." << endl;

}

else {

deleteInt(num1);

}

}

else if (choice == 'N') {

cout << "No integer is deleted." << endl;

}

else {

cout << "Invalid option is chosen" << endl;

outs << "Invalid option is chosen" << endl;

}

}

cout << endl;

cout << "------------------------------------------------------------------------------------------" << endl;

cout << " Thank You For Using Our Software " << endl;

cout << "------------------------------------------------------------------------------------------" << endl;

ins.close();

outs.close();

system("pause");

return 0;

}

# Initial Test Plan

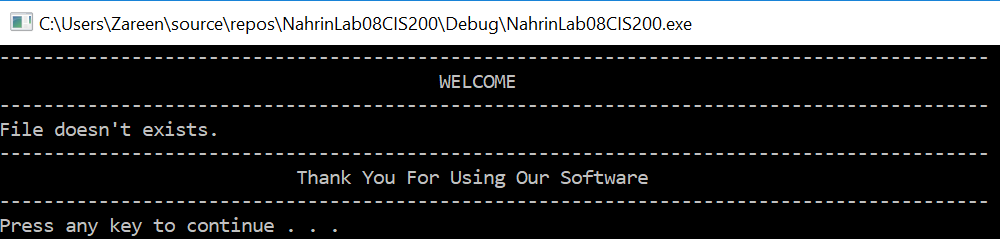
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test # | Valid / Invalid Data | Description of test | Input Value | Expected Output | Actual Output | Test Pass / Fail |
| 1 | Invalid | When the file “integer.dat” does not exists | Filename = “integer.dat” | File doesn’t exists.  “errorlog.txt” output:  File doesn’t exists |  |  |
| 2 | Invalid | When the file does not contain any integer | Filename = “integer.dat” | Output Screen:  “File is empty”  “errorlog.txt” :  “File is empty” |  |  |
| 3 | Invalid | Some of the integers are less than or equal to zero, and rest are positive integers | File integers:  0 -3 3 1 5 2 6 | For integer 0 and -3, screen output:  Numbers can’t be less than one  Numbers can’t be less than one  “errorlog” output:  Numbers can’t be less than one  Numbers can’t be less than one |  |  |
| 4 | Valid | User wants to print the list in the ascending order | File input:  0 -3 3 1 5 2 6  Choice: A | Full List:  1 2 3 5 6  Even List:  1 3 5 |  |  |
| 5 | Valid | User wants to print the list in the descending order | File input:  3 1 2 5 6 9  Choice: D | List in descending order:  9 6 5 3 2 1 |  |  |
| 6 | Invalid | When user input is not A or D for printing the list | File input:  3 1 2 5 6 9  Choice: g | Invalid option |  |  |
| 7 | Valid | When the user wants to add a value in the list but the value already exists. | File input:  3 1 2 5 6 9  Choice: Y  Value: 3 | Number already exists.  “errorlog.txt” output:  Number already exists. |  |  |
| 8 | Invalid | When the user wants to add a value in the list but the value is less than zero | File input:  3 1 2 5 6 9  Choice: Y  Value: -1 | Numbers can’t be zero or negative  “errorlog.txt” output:  Numbers can’t be zero or negative |  |  |
| 9 | Invalid | When the user wants to add a value in the list but the value is zero | File input:  3 1 2 5 6 9  Choice: Y  Value: 0 | Numbers can’t be zero or negative  “errorlog.txt” output:  Numbers can’t be zero or negative |  |  |
| 10 | Valid | User wants to add a valid number in the list | File input:  3 1 2 5 6 9  Choice: Y  Value: 7  Choice: A | The new list in ascending order:  1 2 3 5 6 7 9 |  |  |
| 11 | Valid | User does not want to add new items | File input:  3 1 2 5 6 9  Choice: N | No new number is added. |  |  |
| 12. | Invalid | User input is not Y or N | File input:  3 1 2 5 6 9  Choice: m | Invalid option chosen  “errorlog.txt” output:  Invalid option chosen |  |  |
| 13 | Invalid | User wants to delete an integer from the list but the integer does not exist in the list | File input:  3 1 2 5 6 9  Choice: Y  Value: 10 | Integer does not exist in the list.  “errorlog.txt” output:  Integer does not exist in the list. |  |  |
| 14 | Valid | User wants to delete an existing integer from the list | File input:  3 1 2 5 6 9  Choice: Y  Value: 3 | The new list is:  1 2 5 6 9 |  |  |
| 15 | Valid | When user does not want to delete any integer | File input:  3 1 2 5 6 9  Choice: N | No integer is deleted. |  |  |
| 16 | Invalid | User input is not Y or N for choosing option | File input:  3 1 2 5 6 9  Choice: k | Invalid option is chosen.  “errorlog.txt” output:  Invalid option is chosen |  |  |

# Final Test Plan

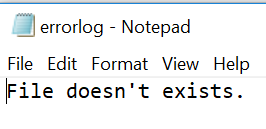
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test # | Valid / Invalid Data | Description of test | Input Value | Expected Output | Actual Output | Test Pass / Fail |
| 1 | Invalid | When the file “integer.dat” does not exists | Filename = “integer.dat” | File doesn’t exists.  “errorlog.txt” output:  File doesn’t exists | File doesn’t exists.  “errorlog.txt” output:  File doesn’t exists | Pass |
| 2 | Invalid | When the file does not contain any integer | Filename = “integer.dat” | Output Screen:  “File is empty”  “errorlog.txt” :  “File is empty” | Output Screen:  “File is empty”  “errorlog.txt” :  “File is empty” | Pass |
| 3 | Invalid | Some of the integers are less than or equal to zero, and rest are positive integers | File integers:  0 -3 3 1 5 2 6 | For integer 0 and -3, screen output:  Numbers can’t be less than one  Numbers can’t be less than one  “errorlog” output:  Numbers can’t be less than one  Numbers can’t be less than one | For integer 0 and -3, screen output:  Numbers can’t be less than one  Numbers can’t be less than one  “errorlog” output:  Numbers can’t be less than one  Numbers can’t be less than one | Pass |
| 4 | Valid | User wants to print the list in the ascending order | File input:  0 -3 3 1 5 2 6  Choice: A | Full List:  1 2 3 5 6  Even List:  1 3 5 | Full List:  1 2 3 5 6  Even List:  1 3 5 | Pass |
| 5 | Valid | User wants to print the list in the descending order | File input:  3 1 2 5 6 9  Choice: D | List in descending order:  9 6 5 3 2 1 | List in descending order:  9 6 5 3 2 1 | Pass |
| 6 | Invalid | When user input is not A or D for printing the list | File input:  3 1 2 5 6 9  Choice: g | Invalid option | Invalid option | Pass |
| 7 | Valid | When the user wants to add a value in the list but the value already exists. | File input:  3 1 2 5 6 9  Choice: Y  Value: 3 | Number already exists.  “errorlog.txt” output:  Number already exists. | Number already exists.  “errorlog.txt” output:  Number already exists. | Pass |
| 8 | Invalid | When the user wants to add a value in the list but the value is less than zero | File input:  3 1 2 5 6 9  Choice: Y  Value: -1 | Numbers can’t be zero or negative  “errorlog.txt” output:  Numbers can’t be zero or negative | Numbers can’t be zero or negative  “errorlog.txt” output:  Numbers can’t be zero or negative | Pass |
| 9 | Invalid | When the user wants to add a value in the list but the value is zero | File input:  3 1 2 5 6 9  Choice: Y  Value: 0 | Numbers can’t be zero or negative  “errorlog.txt” output:  Numbers can’t be zero or negative | Numbers can’t be zero or negative  “errorlog.txt” output:  Numbers can’t be zero or negative | Pass |
| 10 | Valid | User wants to add a valid number in the list | File input:  3 1 2 5 6 9  Choice: Y  Value: 7  Choice: A | The new list in ascending order:  1 2 3 5 6 7 9 | The new list in ascending order:  1 2 3 5 6 7 9 | Pass |
| 11 | Valid | User does not want to add new items | File input:  3 1 2 5 6 9  Choice: N | No new number is added. | No new number is added. | Pass |
| 12. | Invalid | User input is not Y or N | File input:  3 1 2 5 6 9  Choice: m | Invalid option chosen  “errorlog.txt” output:  Invalid option chosen | Invalid option chosen  “errorlog.txt” output:  Invalid option chosen | Pass |
| 13 | Invalid | User wants to delete an integer from the list but the integer does not exist in the list | File input:  3 1 2 5 6 9  Choice: Y  Value: 10 | Integer does not exist in the list.  “errorlog.txt” output:  Integer does not exist in the list. | Integer does not exist in the list.  “errorlog.txt” output:  Integer does not exist in the list. | Pass |
| 14 | Valid | User wants to delete an existing integer from the list | File input:  3 1 2 5 6 9  Choice: Y  Value: 3 | The new list is:  1 2 5 6 9 | The new list is:  1 2 5 6 9 | Pass |
| 15 | Valid | When user does not want to delete any integer | File input:  3 1 2 5 6 9  Choice: N | No integer is deleted. | No integer is deleted. | Pass |
| 16 | Invalid | User input is not Y or N for choosing option | File input:  3 1 2 5 6 9  Choice: k | Invalid option is chosen.  “errorlog.txt” output:  Invalid option is chosen | Invalid option is chosen.  “errorlog.txt” output:  Invalid option is chosen | Pass |

# Screenshots

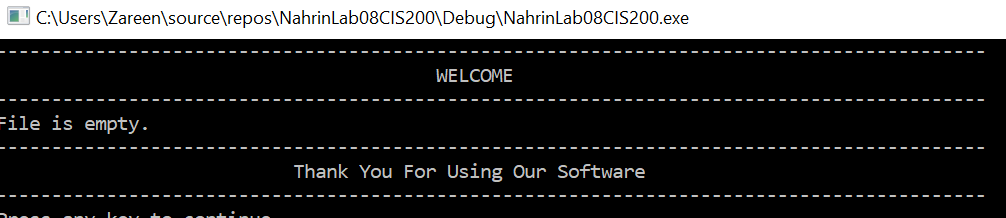
## Test Case 1:



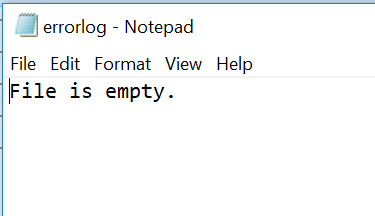
**“errorlog.txt”:**



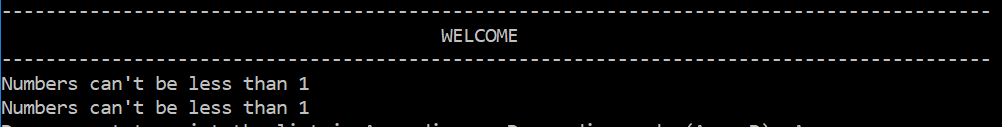
## Test Case 2:



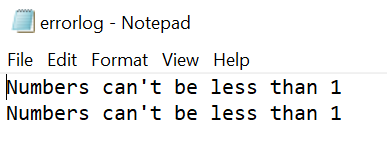
**“errorlog.txt”:**



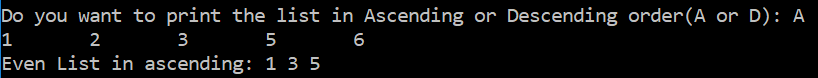
## Test Case 3:



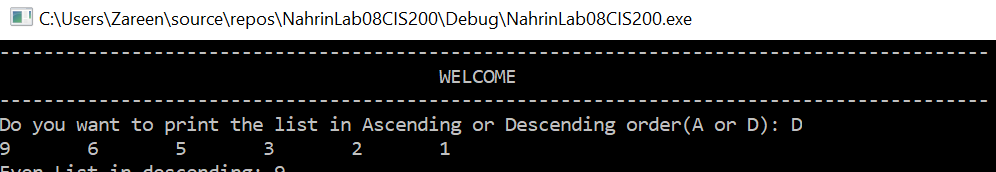
**“errorlog.txt”:**



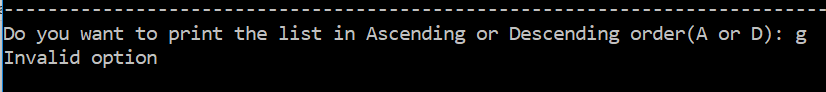
## Test Case 4:



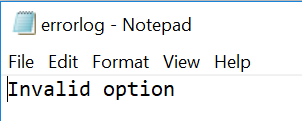
## Test Case 5:



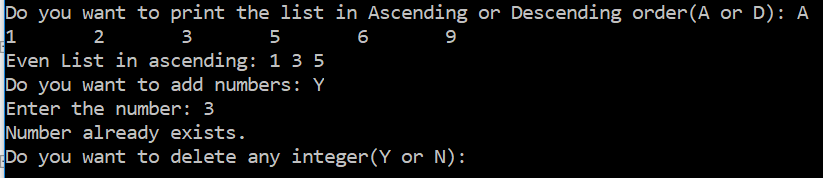
## Test Case 6:



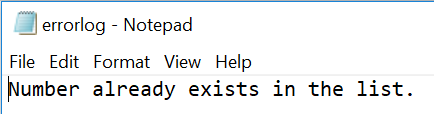
**“errorlog.txt”**



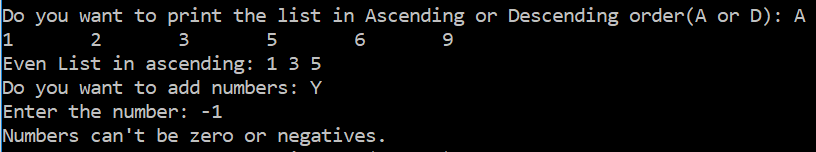
## Test Case 7:



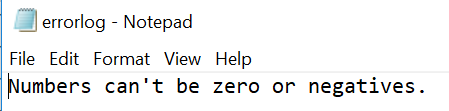
**“errorlog.txt”**



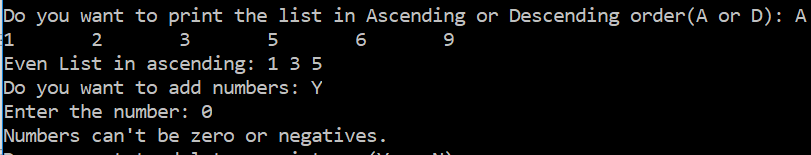
## Test Case 8:



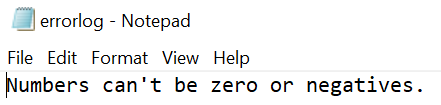
**“errorlog.txt”**



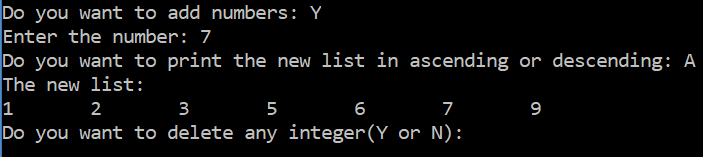
## Test Case 9:



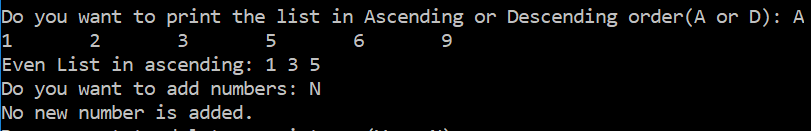
**“errorlog.txt”**



## Test Case 10:



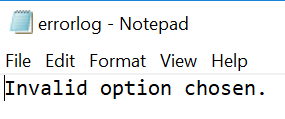
## Test Case 11:



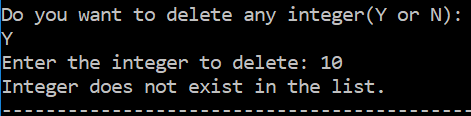
## Test Case 12:



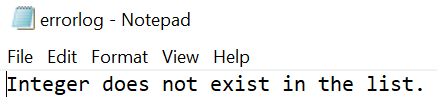
**“errorlog.txt”**



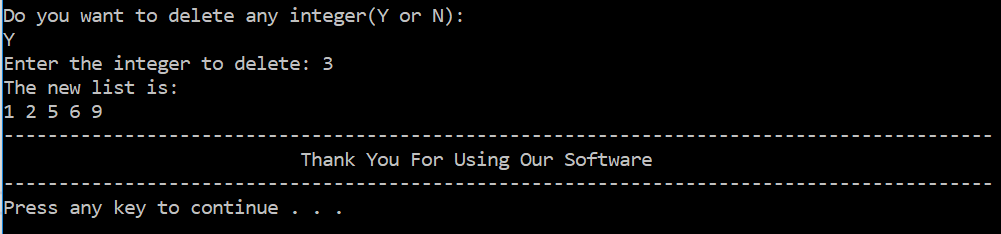
## Test Case 13:



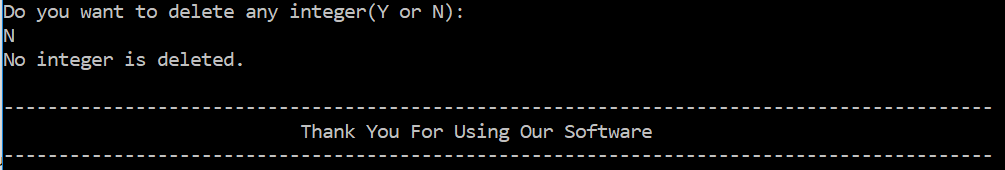
**“errorlog.txt”**



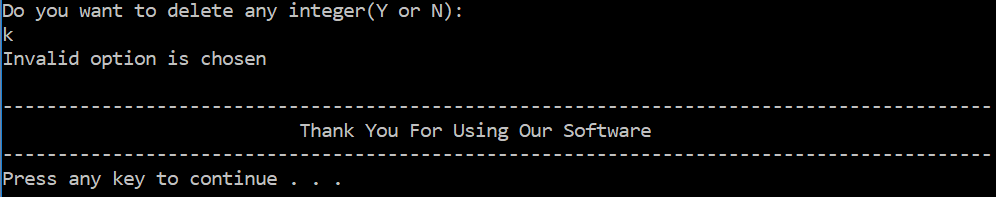
## Test Case 14:



## Test Case 15:



## Test Case 16:



**“errorlog.txt”**

