Programming Assignments CIT 245

**Chapter 1**

Write a program that applies the Babylonian square root algorithm.

The algorithm is:

1. guess = input/2
2. r = input/guess
3. guess = (guess+r)/2
4. go to 2 until done

Example output:

Enter a number and I will apply the Babylonian square root algorithm

until I am withing .001 of the correct answer.

151

You entered 151

guessing 38.75

guessing 21.3234

guessing 14.2024

guessing 12.4172

guessing 12.2889

The Babylons algorithm gives 12.2889

Checking: 12.2889 \* 12.2889 = 151.016

Press any key to continue . . .

**Chapter 2**

Write a program that computes buoyant force given the radius and weight of a sphere. Buoyant Force = Volume x Water Weight (62.4 lbs/cubic foot). Your program should decide whether a sphere sinks or floats in water. The user inputs the radius of the sphere and the weight. Volume = 4/3 pi r^3. Use const double **PI** = 3.141592; for PI.

Your program must prompt for multiple calculations.

Example Output:

This program computes Buoyant Force in water given sphere radius.

Based on the weight of the sphere, it determines whether the sphere floats or sinks.

Enter the radius of the sphere.

1

You entered 1

Enter the weight of the sphere.

1

You entered 1

Buoyant Force = 261.38

Egads, it floats!

Recalculate? (1 = yes, 0 = exit)

1

Enter the radius of the sphere.

1

You entered 1

Enter the weight of the sphere.

400

You entered 400

Buoyant Force = 261.38

It sunk...

Recalculate? (1 = yes, 0 = exit)

0

Press any key to continue . . .

**Chapter 3**

Write a program that uses a function that returns a number between 1 and 6. Use this function to simulate the roll of a die. Allow the user to specify the number of trials and then tabulate that number of rolls of two dice. The program must allow for repeated simulations. Initialize rand() using srand() and time().

Possible Outcomes for any roll

11 12 13 14 15 16

21 22 23 24 25 26

31 32 34 35 36 37

41 42 43 44 45 46

51 52 53 54 55 56

61 62 63 64 65 66

Possible Totals

2 3 4 5 6 7

3 4 5 6 7 8

4 5 6 7 8 9

5 6 7 8 9 10

6 7 8 9 10 11

7 8 9 10 11 12

Therefore, for example, the odds of a 7 are 6:36 in any given roll.

If you roll 36 times then the expected outcome for a 7 is 6. So odds are that you will get a 7, 6 times. The odds you will get a 2 are 1. Tabulate the difference between the expected outcome (odds) and the random number generator outcome – list this as error.

In general, you should notice a decrease in error with more trials.

Percent Formula: [ ((V2 - V1) / |V1|) \* 100 ]

Example Output:

This program rolls two dice and tabulates the results.

Please enter the number of rolls you want: 36

Sum #Rolled Odds %Error

2: 0 1 100.0000

3: 1 2 50.0000

4: 1 3 66.6667

5: 7 4 75.0000

6: 6 5 20.0000

7: 5 6 16.6667

8: 2 5 60.0000

9: 9 4 125.0000

10: 6 3 100.0000

11: 0 2 100.0000

12: 0 1 100.0000

Try again? (1 = Yes, 0 = Exit)

1

Please enter the number of rolls you want: 360

Sum #Rolled Odds %Error

2: 5 10 50.0000

3: 19 20 5.0000

4: 24 30 20.0000

5: 40 40 0.0000

6: 64 50 28.0000

7: 54 60 10.0000

8: 62 50 24.0000

9: 32 40 20.0000

10: 34 30 13.3333

11: 17 20 15.0000

12: 10 10 0.0000

Try again? (1 = Yes, 0 = Exit)

1

Please enter the number of rolls you want: 36000

Sum #Rolled Odds %Error

2: 1007 1000 0.7000

3: 2037 2000 1.8500

4: 3015 3000 0.5000

5: 3970 4000 0.7500

6: 4964 5000 0.7200

7: 6030 6000 0.5000

8: 4977 5000 0.4600

9: 3959 4000 1.0250

10: 3041 3000 1.3667

11: 2008 2000 0.4000

12: 993 1000 0.7000

Try again? (1 = Yes, 0 = Exit)

1

Please enter the number of rolls you want: 36000000

Sum #Rolled Odds %Error

2: 1000809 1000000 0.0809

3: 2002943 2000000 0.1472

4: 3003377 3000000 0.1126

5: 3998148 4000000 0.0463

6: 4998322 5000000 0.0336

7: 5999486 6000000 0.0086

8: 4999017 5000000 0.0197

9: 3997214 4000000 0.0697

10: 3001276 3000000 0.0425

11: 1998918 2000000 0.0541

12: 1000491 1000000 0.0491

Try again? (1 = Yes, 0 = Exit)

1

Please enter the number of rolls you want: 360000000

Sum #Rolled Odds %Error

2: 10008663 10000000 0.0866

3: 19995846 20000000 0.0208

4: 30002587 30000000 0.0086

5: 39991880 40000000 0.0203

6: 50008270 50000000 0.0165

7: 59998415 60000000 0.0026

8: 50008624 50000000 0.0172

9: 39989290 40000000 0.0268

10: 30003247 30000000 0.0108

11: 19991608 20000000 0.0420

12: 10001571 10000000 0.0157

Try again? (1 = Yes, 0 = Exit)

**Chapter 4**

Write a program that reduces a fraction to lowest terms. Two functions should be implemented. Function gcd should find and return the greatest common denominator and be called as g = gcd(num,den);. Function reduce(num,den); should call gcd to provide the lowest terms. The lowest terms should be returned via reference in num and den.

Pseudo code for gcd is:

**function** gcd(a, b)

**while** b ≠ 0

t = b

b = a **mod** b

a = t

**return** a

Example Output:

This program takes a numerator and denominator and reduces to lowest terms

Enter the numerator

28 35

Enter the denominator

greatest common denominator is 7

your fraction reduced is 4/5

Try Again? (1 = yes, 0 = exit)

1

Enter the numerator

85

Enter the denominator

1210

greatest common denominator is 5

your fraction reduced is 17/242

Try Again? (1 = yes, 0 = exit)

1

Enter the numerator

289

Enter the denominator

5148

greatest common denominator is 1

your fraction reduced is 289/5148

Try Again? (1 = yes, 0 = exit)

1

Enter the numerator

306

Enter the denominator

697

greatest common denominator is 17

your fraction reduced is 18/41

Try Again? (1 = yes, 0 = exit)

0

**Chapter 5**

Write a program that generates random numbers between 1 and 10 and fill an array of size 20 with them. Have your program sort the array then output in order the number of occurrences of each random number generated. Use the STL sort function to sort your array.

How to use the STL sort:

#include <algorithm>

...

int a[20]

...

sort(a,a+20);

I recommend pseudo code to develop an algorithm before coding this.

Example Output:

This program generates random numbers and tabulates the results

Original List Sorted:

a[ 0] 1

a[ 1] 1

a[ 2] 1

a[ 3] 2

a[ 4] 2

a[ 5] 3

a[ 6] 6

a[ 7] 6

a[ 8] 6

a[ 9] 7

a[10] 8

a[11] 8

a[12] 8

a[13] 9

a[14] 9

a[15] 9

a[16] 9

a[17] 10

a[18] 10

a[19] 10

N Count

1: 3

2: 2

3: 1

4: 0

5: 0

6: 3

7: 1

8: 3

9: 4

10: 3

Try Again? (1 = yes, 0 = exit)

1

a[ 0] 1

a[ 1] 3

a[ 2] 3

a[ 3] 4

a[ 4] 4

a[ 5] 5

a[ 6] 5

a[ 7] 5

a[ 8] 5

a[ 9] 6

a[10] 6

a[11] 6

a[12] 8

a[13] 9

a[14] 9

a[15] 9

a[16] 9

a[17] 9

a[18] 10

a[19] 10

N Count

1: 1

2: 0

3: 2

4: 2

5: 4

6: 3

7: 0

8: 1

9: 5

10: 2

Try Again? (1 = yes, 0 = exit)

0

Press any key to continue . . .

**Chapter 6**

Write a class that represents a player in a game. The player should have a name, password, experience points, an inventory array of four strings, and a location x,y. Your class should have mutator and accessor methods for all variables. For example: setName(), getName(). It should have a suitable display method. Mutators and accessors should be public but all variables should be private. To implement get inventory, use string \* getInv(); Use the scope resolution operator to implement larger methods such as display(). Use in class methods for shorter methods such as setName(), getName().

Example:

void setName(string name){this->name = name;}

string \*getInv();

...

string\* player::getInv(){return inventory;}

Note that in the above method setName “this->” is required for disambiguation. If written

void setName(string n){name = n;}

“this->” is not required.

Write a test program that creates three players and displays them.

Example Output:

This program generates three player objects and displays them.

Player Info -

Name: Nematocyst

Password: obfuscator

Experience: 1098

Inventory:

Position: 55689, 76453

sword

shield

food

potion

Player Info -

Name: Omphaloskeps

Password: obstreperous

Experience: 11456

Inventory:

Position: 12, 5108

sword of magic

shield of power

mana potion

anti-fire potion

Player Info -

Name: Tokamakfusion

Password: toroid

Experience: 15678

Inventory:

Position: 99, 108

axe of damage

shield of warding

strength potion

healing potion

Press any key to continue . . .

**Chapter 7**

Create a counter. The default constructor should set the count to zero. The overloaded constructor should initialize to any integer. Provide a display(), inc() and dec() method. Inc does ++ dec does – – on the private count variable. Use private: and public: where they should be used.

Write a demo program that initializes one counter to 0 and the other to 200. Increment the first counter by

85 and decrement the second counter by 242. Use a for loop to increment and decrement.

Use a static variable to keep track of the total number of incs and decs. Keep in mind that after you declare a static variable in a class, you must declare it outside the class and use the scope resolution operator.

Example:

Current state of C1 (created with the default constructor)

Current count: 0

total increments and decrements 0

Current state of C2 (created as C2(200))

Current count: 200

total increments and decrements 0

Current state of C1 after 85 inc()

Current count: 85

total increments and decrements 85

Current state of C2 after 242 dec()

Current count: -42

total increments and decrements 327

Press any key to continue . . .

**Chapter 8**

Implement a class money. You must have friend functions for I/O that overload << and >>. You must have multiple constructors, You must implement a percentage method,+,-, <,>. There is a starting money class in your text.

Example:

Enter an amount of money: $14.33

Your amount is $14.33

My amount is $10.09

One of us is richer.

You have more money than me.

10% of your money is: $$1.43

$14.33 + $10.09 equals $24.42

$14.33 - $10.09 equals $4.24

Press any key to continue . . .

**Chapter 9**

Use an STL stack to reverse a line of input characters that are read into a string. Your stack should contain the characters of the user's string. Use getline() for input – it needs to be part of your C++ tool inventory.

A note on getline: Suppose I am doing the following -

This program reverses a string using the STL stack

Enter your string of less than 80 characters followed by an ENTER

a string input

Enter another? 1 = continue. Anything else to stop

1

Enter your string of less than 80 characters followed by an ENTER

a second string

This code will not work by simply using the following loop:

int go = 1;

cout << "This program reverses a string using the STL stack" << endl;

while(go == 1){

cout << "Enter your string of less than 80 characters followed by an ENTER" << endl;

char\* s = (char \*)malloc(80);

cin.getline(s,81,'\n');

cout << s << endl;

cout << "Enter another? 1 = continue. Anything else to stop" << endl;

cin >> go;

}

Try it and see what happens! Also note that I never got rid of the memory I allocated with malloc – your code must get rid of it. Also, malloc outside the loop for more efficient code.

You must use a getchar() (part of the cstdio library) after cin >> go;

The reason is that when you enter 1 you also use the Enter key. This is still in the buffer when you hit getline again! So you will read in '\n'

Also note that when you get something off of the STL stack you must use .top() to look at it followed by .pop() to remove it.

Example:

This program reverses a string using the STL stack

Enter your string of less than 80 characters followed by an Enter

m&m cheeto mayo

oyam oteehc m&m

Enter another? 1 = continue. Anything else to stop

1

Enter your string of less than 80 characters followed by an Enter

celery lettuce apple

elppa ecuttel yrelec

Enter another? 1 = continue. Anything else to stop

0

Press any key to continue . . .

**Chapter 10** Pointers

Part 1

Create an int i with a value of 7. Create a pointer to integer pi. Point your pointer pi to your int variable i. Print out your pointer, the address of your pointer and a dereference of your pointer.

Create a pointer to your integer pointer ppi. Point it to your pointer to int pi. Print out ppi, the address of ppi, a dereference to ppi and a double dereference to ppi.

Part 2

Understanding deep vs. shallow copy is essential for a programmer. You will get into severe problems trying to code if you do not understand it.

The essence of the problem is that 2 objects, which should have independent memory storage, accidently wind up sharing memory.

I want you to wrap a character array (and array with 'a','b','c','d','e' is strictly speaking not a string since it does not end in '\0') with a class (this is just a class that contains an array) and then properly(in Deep) and improperly (in Shallow) assign memory.

Make a class WrapArrayDeep that has a private pointer to char. Your default constructor should allocate an array of size 5. You should have a copy constructor that does a deep copy. (allocates a new array)

Your WrapArrayDeep class should start like:

class WrapArrayDeep{

      char \*pch;

      WrapArrayDeep(){

            pch = new char[5];

            \*pch = 97; //etc.

      }

      WrapArrayDeep(WrapArrayDeep wad){

           // correct copy constructor.

      }

}

Make a similar class, WrapArrayShallow, that has an improper copy constructor that causes your copy to point to the array in the source object. (instead of making a new array, have pch point to the original array)

Demonstrate the difference between the classes use

WrapArrayDeep wad1, \*wad2;

for the variables holding your WrapArrayDeeps and for WrapArrayShallow:

WrapArrayShallow was1, \*was2;

Be sure to include a destructor in each class – note it must be an ARRAY destructor put a cout in the destructor showing it was called..

In WrapArrayDeep:

Use pointer arithmetic to load your array with ASCII values for letters.

\*pca = 97;

\*(pca+1) = 98;

etc.

Use array notation to print your array.

for(int I = 0; I < 5; I++)

cout << pca[i] << endl;

In WrapArrayShallow:

Use array notation to load your array with char data.

pca[0]='v';

pca[1]='w';

etc

Use pointer arithmetic to print your array.

for(int I = 0; I < 5; i++)

cout << \*(pca + 1) << endl;

Example Output:

this program section uses 3 variables

i = 7, pi a pointer to i and ppi a pointer to pi

pi = 002EF738

dereference pi 7

address of pi 002EF744

address of i 002EF738

ppi = 002EF744

dereference of ppi 002EF738

address of ppi 002EF72C

double dereference of ppi

7

this section instantiates a wrapper class for a dynamic array of 5 elements

WrapArrayDeep 1

a b c d e

WrapArrayDeep 2 created using the copy constructor on 1

a b c d e

after changing the contents of WrapArrayDeep 1, 1 and 3 =

{ | } ~ ⌂

a b c d e

Now doing the same thing with WrapArrayShallow

wrapArrayShallow 1

a b c d e

wrapArrayShallow 2 created using the copy constructor on 1

a b c d e

after changing the contents of WrapArrayShallow 1, 1 and 3 =

{ | } ~ ⌂

{ | } ~ ⌂

calling destructor for WrapArrayShallow

calling destructor for WrapArrayShallow  **\*\*\*\*\* this may or may not work depending on your compiler**

calling destructor for WrapArrayDeep

calling destructor for WrapArrayDeep

Press any key to continue . . .

**\*\*\*\*\* If this crashes your program simply remove it.**

**Chapter 11**

Create 5 files. Driver.cpp, f.h, f.cpp, g.h, g.cpp. f and g should implement a function called hello. Driver should call hello from f and g.

Example Output:

hello from f

hello from g

Press any key to continue . . .

Zip your source files before submitting.

**Chapter 12**

Make a program that can write advice to a file. Your program should not require a data file to be present when it is run. If a file needs to be created, your program should do so.

According to ANSI standards, all compilers should operate the same way. Naturally they do not. David Busch one of my Spring 2011 students, managed to come up with code that worked with both MSVC++ and DevC++.

In MSVC++ his code:

inStream.open("advice.txt");

When no file is present, passes the test:

// Does this file exist?

if (!inStream.fail()) // file exists...

Because MS has decided to create a file if none exists. I believe this is not ANSI standard but is more convenient for a programmer.

For DevC++, his code:

if (!inStream.fail()){ // file exists…

here is code to read in the orginal advice. Of course in MSVC++ the file is empty.

}

else{ // This file does NOT exist.

inStream.close(); // close the input file stream

inOutStream.open("advice.txt", ios::in | ios::out | ios::trunc); // create new file

}

Handles the case where no file exists and one must be created. I suggest using this approach since it is compiler independent.

Important: Note that there are 3 types of file streams, an ofstream only does output, an ifstream only does input. If you want input and output on the same file you use an fstream. inStream.open("advice.txt",ios::app); will always cause an error if you declared ifstream inStream; Unfortunately, it will compile but will always fail at run time because you are trying to append to an input stream.

Example Output:

First Run (Dev C++):

Could not open Advice File.

Assumption: first run - creating a new file...

Enter your phrase for the next user:

Never take advice from a programmer.

Press any key to continue . . .

Second Run(Dev C++)

Found Advice File.

Old Advice:

Never take advice from a programmer.

Enter your phrase for the next user:

Ok I never will! Ooops, I just did.

Press any key to continue . . .

**Chapter 13**

Research and implement the towers of Hanoi. Use a recursive rather than iterative solution.

Example Output:

enter number of disks

3

source 1 target 2 temporary 3

from 1 to 2

from 1 to 3

from 2 to 3

from 1 to 2

from 3 to 1

from 3 to 2

from 1 to 2

2 to the 3 power = 8

Number of moves 7

Continue? (1=yes 0=no)

**Chapter 14**

You should know inheritance from Java but to get a chance to use C++ syntax, make a base class gun. Gun should have

string manufacturer;

string caliber;

handgun should inherit from gun and add

string grips;

string sights;

bool laser;

Finally have classes revolver and pistol that inherit from handgun. Revolver should add:

bool singleAction;

int numberOfRounds;

Pistol should have:

bool semiauto;

Make good decisions about constructors, private variables, mutators and accessors.

Make an appropriate test program by instantiating and printing at least one revolver and one pistol.

Example Output:

Revolver: Smith & Wesson, 0.357, 6 shot, double action, hoague grips, laser, Trijicon

Pistol: Glock, 9mm, semi-auto, manufacturers grips, no laser, 3 dot sights

**Chapter 15**

Add a virtual print function to your base class gun in your chapter 15 homework.

Make an array gunCabinet that contains revolvers and pistols and prints out its contents using the following loop:

for(i=0; i < numGuns; i++)

gunCabinet[i]->print();

Example Output:

My Gun Cabinet Contains:

Revolver: Smith & Wesson, 0.357, 6 shot, double action, hoague grips, laser, Trijicon

Pistol: Glock, .40, semi-auto, synthetic grips, no laser, 3 dot sights

**Chapter 16**

Template a binary search function. Prove that it works with at least 2 types of data.

Example Output:

Integer test array contains:

0 1 1 2 3 5 6 13 21 32 55

-1 is not in the array.

0 is at index 0

1 is at index 2

2 is at index 3

3 is at index 4

4 is not in the array.

5 is at index 5

6 is at index 6

7 is not in the array.

String test array contains:

head knees shoulders toes

elbows is not in the array.

knees is at index 1

**Chapter 17**

Take the code I have written here and change it into a doubly linked list. You cannot write the doubly linked list without understanding the singly linked list presented. There are tons of good tutorials on singly and doubly linked lists online.

/\*

Written by Professor Kenneth L Moore

For CIT245 Data Structures and Programming C++

This code implements a singly linked list

Students must change it to a doubly linked list

\*/

#include <iostream>

#include <string>

using namespace std;

// define a node for storage and linking

class node{

public:

string name;

node \*next;

// node \*prev; // to be implemented by students

};

class linkedList{

public:

linkedList():top(NULL){}

bool empty(){return top == NULL;}

node \*getTop(){return top;}

void setTop(node \*n){top = n;}

void add(string);

int menu();

void remove(string);

~linkedList();

// void reversePrint(); // to be implemented by students

friend ostream& operator << (ostream&, const linkedList&); // default output is in-order print.

private:

node \*top;

//node \*end; // to be used for reverse print and implemented by students

};

void main(){

linkedList l;

cout << l.empty() << endl;

int option = 0;

string s;

bool go = true;

while(go){

option = l.menu();

switch(option){

case 1: cout << "enter a name: ";cin >> s; l.add(s); break;

case 2: cout << "enter name to be deleted: "; cin >> s; l.remove(s);break;

case 3: cout << l; break;

case 4: cout << "can not be done with a singly linked list"<< endl;

case 5: cout << "exiting" << endl; go = false; break;

}

}

// l goes out of scope and calls ~linkedList()

}

// can not call this method "delete" - "delete" is a reserved keyword.

void linkedList::remove(string s){

bool found = false;

node \*curr = getTop(), \*prev=NULL;

while(curr != NULL){

// match found, delete

if(curr->name == s){

found = true;

// found at top

if(prev == NULL){

node \*temp = getTop();

setTop(curr->next);

delete(temp);

// found in list - not top

}else{

prev->next = curr->next;

delete(curr);

}

}

// not found, advance pointers

if(!found){

prev = curr;

curr = curr->next;

}

// found, exit loop

else curr = NULL;

}

if(found)cout << "Deleted " << s << endl;

else cout << s << " Not Found "<< endl;

}

void linkedList::add(string s){

node \*n = new node();

n->name = s;

n->next = NULL;

// take care of empty list case

if(empty()){

top = n;

// take care of node belongs at beginning case

} else if(getTop()->name > s){

n->next = getTop();

setTop(n);

// take care of inorder and end insert

}else{

// insert in order case

node \*curr = getTop(), \*prev = curr;

while(curr != NULL){

if(curr->name > s)break;

prev = curr;

curr = curr->next;

}

if(curr != NULL){ // search found insert point

n->next = curr;

prev->next = n;

}

// take care of end of list insertion

else if(curr == NULL){// search did not find insert point

prev->next = n;

}

}

}

ostream& operator << (ostream& os, const linkedList& ll){

//linkedList x = ll; // put this in and the code blows up - why?

node \*n = ll.top;

if(n == NULL)cout << "List is empty." << endl;

else

while(n != NULL){

os << n->name << endl;

n = n->next;

}

return os;

}

// return memory to heap

linkedList::~linkedList(){

cout << "~linkedList called." << endl;

node \*curr = getTop(), \*del;

while(curr != NULL){

del = curr;

curr = curr->next;

delete(del);

}

}

int linkedList::menu(){

int choice = 0;

while(choice < 1 || choice > 5){

cout << "\nEnter your choice" << endl;

cout << " 1. Add a name." << endl;

cout << " 2. Delete a name." << endl;

cout << " 3. Show list." << endl;

cout << " 4. Show reverse list. " << endl; // to be implemented by students

cout << " 5. EXIT " << endl;

cin >> choice;

}

return choice;

}

Example Output:

1

Enter your choice

1. Add a name.

2. Delete a name.

3. Show list.

4. Show reverse list.

5. EXIT

1

enter a name: liz

Enter your choice

1. Add a name.

2. Delete a name.

3. Show list.

4. Show reverse list.

5. EXIT

1

enter a name: bonnie

Enter your choice

1. Add a name.

2. Delete a name.

3. Show list.

4. Show reverse list.

5. EXIT

1

enter a name: al

Enter your choice

1. Add a name.

2. Delete a name.

3. Show list.

4. Show reverse list.

5. EXIT

1

enter a name: zeek

Enter your choice

1. Add a name.

2. Delete a name.

3. Show list.

4. Show reverse list.

5. EXIT

1

enter a name: willy

Enter your choice

1. Add a name.

2. Delete a name.

3. Show list.

4. Show reverse list.

5. EXIT

1

enter a name: wanda

Enter your choice

1. Add a name.

2. Delete a name.

3. Show list.

4. Show reverse list.

5. EXIT

3

al

bonnie

liz

wanda

willy

zeek

Enter your choice

1. Add a name.

2. Delete a name.

3. Show list.

4. Show reverse list.

5. EXIT

4

zeek

willy

wanda

liz

bonnie

al

Enter your choice

1. Add a name.

2. Delete a name.

3. Show list.

4. Show reverse list.

5. EXIT

2

enter name to be deleted: liz

Deleted liz

Enter your choice

1. Add a name.

2. Delete a name.

3. Show list.

4. Show reverse list.

5. EXIT

2

enter name to be deleted: al

Deleted al

Enter your choice

1. Add a name.

2. Delete a name.

3. Show list.

4. Show reverse list.

5. EXIT

2

enter name to be deleted: zeek

Deleted zeek

Enter your choice

1. Add a name.

2. Delete a name.

3. Show list.

4. Show reverse list.

5. EXIT

2

enter name to be deleted: zeek

zeek Not Found

Enter your choice

1. Add a name.

2. Delete a name.

3. Show list.

4. Show reverse list.

5. EXIT

3

bonnie

wanda

willy

Enter your choice

1. Add a name.

2. Delete a name.

3. Show list.

4. Show reverse list.

5. EXIT

2

enter name to be deleted: bonnie

Deleted bonnie

Enter your choice

1. Add a name.

2. Delete a name.

3. Show list.

4. Show reverse list.

5. EXIT

2

enter name to be deleted: willy

Deleted willy

Enter your choice

1. Add a name.

2. Delete a name.

3. Show list.

4. Show reverse list.

5. EXIT

2

enter name to be deleted: wanda

Deleted wanda

Enter your choice

1. Add a name.

2. Delete a name.

3. Show list.

4. Show reverse list.

5. EXIT

3

List is empty.

Enter your choice

1. Add a name.

2. Delete a name.

3. Show list.

4. Show reverse list.

5. EXIT

1

enter a name: tawnya

Enter your choice

1. Add a name.

2. Delete a name.

3. Show list.

4. Show reverse list.

5. EXIT

3

tawnya

Enter your choice

1. Add a name.

2. Delete a name.

3. Show list.

4. Show reverse list.

5. EXIT

5

exiting

~linkedList called.

Press any key to continue . . .

**Chapter 19: STL Assignment**

Create a map and store social security numbers by names.

Iterate through your container and retrieve all names.

Seach for an individual name and print the ssn.

Search for a name that is not found.

Example Output:

Person 123-45-6790: Paul Brown

Person 123-45-6791: Mary Smith

Person 123-45-6789: John Smith

Person 123-45-6792: Lisa Brown

Iterating through list...

123456789: John Smith

123456790: Paul Brown

123456791: Mary Smith

123456792: Lisa Brown

John Brown not found

Paul Brown found 12345790

Press any key to continue . . .