Lab 3: Pointers

CSC 2052/2054

Parts 1-3 are about understanding syntax and the difficulty is how to write the code, part 4 is using pointers in a program.

Part I:

a. create 2 ints ,aInt and bInt, and 2 int pointers, aPtr and bPtr, set each int pointer to the corresponding int.

b. using the pointers, set aInt to 8 and bInt to 11. Print out the values of aInt and bInt using the pointers and label each. Also print the memory address each pointer holds (i.e. print aPtr and bPtr).

c. flip the two pointers so aPtr points to bInt and bPtr points to aInt. Use only pointers for this part of your code; do not use aInt or bInt. You can use as many new pointers as you think you will need.

d. print out the values and memory addresses again.

Sample run: ( 0076FAF4 and 0076FAE8 are my memory addresses; yours will be different)

aptr: 8

bptr: 11

aptr: 0076FAF4 bptr: 0076FAE8

aptr: 11

bptr: 8

aptr: 0076FAE8 bptr: 0076FAF4

\_\_\_\_\_\_\_ Demonstrate that part I runs correctly. (10 points)

Part II: [15 points]

a. Create a vector of 26 char \*s. Instantiate each one to a letter of the alphabet. a should be at index 0, b at index 1, …, z at index 25. Use a for loop rather than setting each of the chars by hand. Use the push\_back function to add your char to the vector.   
 Hint: recall ‘a’ + 0 is ‘a’, ‘a’+1 is ‘b’ , ‘a’ + 2 is ‘c’, etc.

b. Print out each letter.

c. Delete each char \*. (print and delete should happen in different for loops)

Output:

a

b

c

d

e

f

g

h

i

j

k

l

m

n

o

p

q

r

s

t

u

v

w

x

y

z

Part III: [15 points]

Create a 2d array of size 15 by 15, using dynamic allocation.

Set each spot in the array to be the addition of the indexes. Print each value out in the way listed below. Delete the array afterward.

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

3 4 5 6 7 8 9 10 11 12 13 14 15 16 17

4 5 6 7 8 9 10 11 12 13 14 15 16 17 18

5 6 7 8 9 10 11 12 13 14 15 16 17 18 19

6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

7 8 9 10 11 12 13 14 15 16 17 18 19 20 21

8 9 10 11 12 13 14 15 16 17 18 19 20 21 22

9 10 11 12 13 14 15 16 17 18 19 20 21 22 23

10 11 12 13 14 15 16 17 18 19 20 21 22 23 24

11 12 13 14 15 16 17 18 19 20 21 22 23 24 25

12 13 14 15 16 17 18 19 20 21 22 23 24 25 26

13 14 15 16 17 18 19 20 21 22 23 24 25 26 27

14 15 16 17 18 19 20 21 22 23 24 25 26 27 28

Part IV: [40 points]

a. Create a vector of 10 int pointers. Instantiate each of them. Recall instantiate means to allocate.

b. In a while loop, present the user with a menu to do the following (the menu should be in a separate function, printmenu) depending on a char they enter. Print menu should only print the menu.

if s - set a value in the vector to an input value. User input Format: INDEX VALUE [first you enter the index then you enter the value]

if p - print a value of the vector at an index, input by the user. Input Format: INDEX

if d - delete the memory at an index. Input Format: INDEX

if i – instantiate (call new) the memory at an input index. (do not set the memory you instantiated to a value) Input Format: INDEX

if w – add two values by calling a function, add, that takes in two of the pointers from your array as arguments, selected by inputting two indexes, and adds the values of the pointers (the values, not the memory addresses) and sets both values to be the added value Input Format: INDEX INDEX   
 (as an example) Pointers can be passed to functions as in the following function prototype: void print(int \* ptrParam, int intparam). ptrParam is an int pointer that is passed into the function. Like passing by reference, if you change the value of a pointer in a function, it changes outside of the function.

if q – quit. On quit it should delete all the memory.

c. Add code to prevent the user from leaking memory (calling new on an index that already has valid memory). Add code to prevent the program from crashing due to using or accessing null or deleted memory. Add code to prevent index out of ranges. In the case of the example run, “index 1 is null” was printed because the pointer 1 was deleted and then printed before it was set again. The checks should be done AFTER all the input is asked for. Hint: *What did Dr. Mood suggest you should always do to pointers once you deleted what it contained? Think you can check for that with if(mypointer== <something>)?*

Sample run.

Menu:

s: set a value

p: print a value

d: delete a value

i: create a new int

w: add two values

q: quit

: ->> s

->> 9

->> 100

Menu:

s: set a value

p: print a value

d: delete a value

i: create a new int

w: add two values

q: quit

: ->> p

->> 9

100

Menu:

s: set a value

p: print a value

d: delete a value

i: create a new int

w: add two values

q: quit

: ->> s

->> -1

->> 10

Index is out of range!

Menu:

s: set a value

p: print a value

d: delete a value

i: create a new int

w: add two values

q: quit

: ->> s

->> 10

->> 10

Index is out of range!

Menu:

s: set a value

p: print a value

d: delete a value

i: create a new int

w: add two values

q: quit

: ->> i

->> 1

index 1 is active memory

Menu:

s: set a value

p: print a value

d: delete a value

i: create a new int

w: add two values

q: quit

: ->> d

->> 1

Menu:

s: set a value

p: print a value

d: delete a value

i: create a new int

w: add two values

q: quit

: ->> d

->> 1

index 1 is null

Menu:

s: set a value

p: print a value

d: delete a value

i: create a new int

w: add two values

q: quit

: ->> i

->> 1

Menu:

s: set a value

p: print a value

d: delete a value

i: create a new int

w: add two values

q: quit

: ->> s

->> 8

->> 5

Menu:

s: set a value

p: print a value

d: delete a value

i: create a new int

w: add two values

q: quit

: ->> w

->> 8

->> 9

Menu:

s: set a value

p: print a value

d: delete a value

i: create a new int

w: add two values

q: quit

: ->> q

\_\_\_\_\_\_\_ Demonstrate parts II, III, and IV of the program. (70 points)

Additional Questions (20 points):

(2 points) 3.2

(2 points) 3.3

(2 points) 3.4

(2 points) 3.6

(3 points) 3.12

(2 points) 3.15

(7 points) 3.18