**Name (s): Kaumil Patel**

**Course Name:** Principles of Software Design

**Lab Section:** B01 (with Dr. Moussavi)

**Course Code:** ENSF 480

**Assignment Number:** Lab 1

**Submission Date and Time:** 23/09/2021

**Exercise A (12 marks) – Review of C++ Fundamentals Learned in ENSF 337**

|  |  |
| --- | --- |
| Program output and its order | Your explanation (why and where this output was created) |
| constructor with int argument is called | It is called at line 12 in exAmain.  The statement, Mystring c = 3; is interpreted by the compiler as a call to the constructor Mystring::Mystring(int n). |
| default constructor is called  default constructor is called | At line 18 in exAmain.  The statement, Mystring x[2]; calls the default constructor 2 time to initialize the array. |
| constructor with char\* argument is called | At line 22 in exAmain.  In statement, Mystring\* z = new Mystring("4"); z is initialized with a char array which call the constructor with char\* argument |
| copy constructor is called  copy constructor is called | At line 24 in exAmain.  In statement, x[0].append(\*z).append(x[1]); append is called twice and the argument is a Mystring object. Since it doesn’t ask for a reference of the object. The copy constructor is called and a copy of the object is sent to the function. |
| destructor is called  destructor is called | At line 25 in exAmain.  After statement, x[0].append(\*z).append(x[1]); the destructor is called since the two copies are out of the scope and that memory need to be freed. |
| copy constructor is called | At line 26 in exAmain.  At statement, Mystring mars = x[0]; since mars is being initialized and x[0] is a Mystring object, the copy constructor is called. |
| assignment operator called | At line 28 in exAmain.  At statement, x[1] = x[0]; x[1] is already initialized thus the assignment operator is called |
| constructor with char\* argument is called  constructor with char\* argument is called | At line 30 in exAmain.  The statement, Mystring jupiter("White"); since jupiter is being initialized with a char array, the constructor with char\* argument is called  At line 32 in exAmain.  At statement, ar[0] = new Mystring ("Yellow"); a[0] is being assigned a new mystring object with a char array as the argument therefore the constructor with char\* argument is called |
| destructor is called  destructor is called  destructor is called  destructor is called  destructor is called | At line 34 in exAmain.  At line 34 in exAmain.  At line 34 in exAmain.  At line 34 in exAmain.  The block ended and the variables declared inside are out of scope there the destructor is called for each of them. x[0], x[1], mars, and Jupiter call the destructor.  At line 37 in exAmain.  The statement, delete ar [0]; calls for the delete constructor to free memory at ar[0]. |
| constructor with char\* argument is called | At line 39 in exAmain.  The statement, Mystring d = "Green"; d is initialized with a char array therefore the constructor with char\* argument is called |
| Program terminated successfully | At line 41 in exAmain.  The statement cout << "\nProgram terminated successfully." <<endl; before the program terminates. |
| destructor is called  destructor is called | At line 42 in exAmain.  At line 42 in exAmain.  After program ends it cleans memory for remaining objects.  Destructor is called on d and c |

**Exercise B (14 marks):**

**Part I – Drawing an AR Diagram for a Dictionary Data Structure (5 marks):**

AR diagram for **point two** in function remove, when the program reaches this point for the first time.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Stack**   |  |  |  |  | | --- | --- | --- | --- | | **DictionaryList::remove** | |  |  | | --- | --- | | **doomed\_node** |  | | | |  |  | | --- | --- | | **keyA** |  |      |  |  | | --- | --- | | **this** |  | |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | **dictionary\_tests** | |  |  | | --- | --- | | **temp** | **8002** |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | **dl** | |  |  | | --- | --- | | **sizeM** | **3** |  |  |  | | --- | --- | | **cursor** | **0** |  |  |  | | --- | --- | | **headM** |  | | | | **No args** |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | **main** | |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | **dl** | |  |  | | --- | --- | | **sizeM** | **?** |  |  |  | | --- | --- | | **cursor** | **?** |  |  |  | | --- | --- | | **headM** | **?** | | | | **No args** | | **Heap**   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | **‘D’** | **‘i’** | **‘l’** | **‘b’** | **‘e’** | **‘r’** | **‘t’** |  |  |  |  |  |  | | --- | --- | --- | --- | --- | | **‘A’** | **‘l’** | **‘i’** | **‘c’** | **‘e’** |  |  |  |  |  |  | | --- | --- | --- | --- | --- | | **‘W’** | **‘a’** | **‘l’** | **‘l’** | **‘y’** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | |  | | --- | | **datumM** | | |  |  | | --- | --- | | **charsM** |  |  |  |  | | --- | --- | | **length** | **7** | |  |  |  | | --- | --- | | **keyM** | **8001** |      |  |  | | --- | --- | | **nextM** |  | | | |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | |  | | --- | | **datumM** | | |  |  | | --- | --- | | **charsM** |  |  |  |  | | --- | --- | | **length** | **5** | |  |  |  | | --- | --- | | **keyM** | **8002** |      |  |  | | --- | --- | | **nextM** |  | | | |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | |  | | --- | | **datumM** | | |  |  | | --- | --- | | **charsM** |  |  |  |  | | --- | --- | | **length** | **5** | |  |  |  | | --- | --- | | **keyM** | **8003** |      |  |  | | --- | --- | | **nextM** |  | | |      |  | | --- | | **NULL** | |

**Part II – Writing Missing Functions (9 marks):**

Printing list just after its creation ...

List is EMPTY.

Printing list after inserting 3 new keys ...

8001 Dilbert

8002 Alice

8003 Wally

Printing list after removing two keys and inserting PointyHair ...

8003 Wally

8004 PointyHair

Printing list after changing data for one of the keys ...

8003 Sam

8004 PointyHair

Printing list after inserting 2 more keys ...

8001 Allen

8002 Peter

8003 Sam

8004 PointyHair

\*\*\*----Finished dictionary tests---------------------------\*\*\*

Printing list--keys should be 315, 319

315 Shocks

319 Randomness

Printing list--keys should be 315, 319, 335

315 Shocks

319 Randomness

335 ParseErrors

Printing list--keys should be 315, 335

315 Shocks

335 ParseErrors

Printing list--keys should be 319, 335

319 Randomness

335 ParseErrors

Printing list--keys should be 315, 319, 335

315 Shocks

319 Randomness

335 ParseErrors

\*\*\*----Finished tests of copying----------------------\*\*\*

Let's look up some names ...

name for 8001 is: Allen.

Sorry, I couldn't find 8000 in the list.

name for 8002 is: Peter.

name for 8004 is: PointyHair.

\*\*\*----Finished tests of finding -------------------------\*\*\*

**void DictionaryList::find(const Key &keyA) {**

**for (Node \*p = headM; p != 0; p = p->nextM) {**

**if (keyA == p->keyM) {**

**cursorM = p;**

**return;**

**}**

**}**

**cursorM = 0;**

**}**

**void DictionaryList::destroy() {**

**sizeM = 0;**

**delete cursorM;**

**while (headM != 0) {**

**cursorM = headM->nextM;**

**delete headM;**

**headM = cursorM;**

**}**

**cursorM = 0;**

**headM = 0;**

**}**

**void DictionaryList::copy(const DictionaryList &source) {**

**if (source.sizeM == 0) {**

**sizeM = 0;**

**cursorM = 0;**

**headM = 0;**

**} else {**

**sizeM = source.sizeM;**

**cursorM = 0;**

**Node \*currSource = source.headM;**

**Node \*curr;**

**curr = new Node(currSource->keyM, currSource->datumM, 0);**

**headM = curr;**

**if (currSource == source.cursorM) {**

**cursorM = curr;**

**}**

**while (currSource->nextM != 0) {**

**curr->nextM = new Node(currSource->nextM->keyM, currSource->nextM->datumM, 0);**

**curr = curr->nextM;**

**currSource = currSource->nextM;**

**if (currSource == source.cursorM) {**

**cursorM = curr;**

**}**

**}**

**}**

**}**

**Exercise C - (10 marks)**

Files in folder

**Exercise D - (15 marks)**

Files in folder