Final Exam

FALL 2021 CSC 461/361

Student Information

Integrity Policy: All university integrity and class syllabus policies have been followed. I have neither given, nor received, nor have I tolerated others' use of unauthorized aid.

I understand and followed these policies: [✓ Yes] No

Name:	Joey Domino
Date:	11/17/2021

Submission Details

Final ***Changelist*** number:	124492
Verified build:	[✓ Yes] No
Required Configuration:	_DEBUG_ only

Discussion (What did you learn):

I learned:

- Nothin! It's time to shit out what I learned man!

- That I'm not done yet! Still need to wrap up the particle system! WOOO

- That I found a pretty great instructor to have as my first teacher at DePaul. Happy Holidays Ed. I enjoyed this class and learning from you. You are insanely smart. I look forward to ripping my hair out more in the new year!

Verify Builds

- Follow the Piazza procedure on submission
    - Verify your submission compiles and works at the changelist number.
- Verify that only MINIMUM files are submitted
    - No – Generated files
        - *.pdb, *.suo, *.sdf, *.user, *.obj, *.exe, *.log, *.pdb, *.db, *.user
        - Anything that is generated by the compiler should not be included
    - No – Generated directories
        - /Debug, /Release, /Log, /ipch, /.vs
- Typical files project files that are required
    - *.sln, *.cpp, *.h
    - *.vcxproj, *.vcxproj.filters, CleanMe.bat

Standard Rules

Write all programs in cross-platform C++

- Optimize for execution speed and robustness
- Working code doesn't mean full credit

Submission Report

- Fill out the submission Report
    - No report, no grade

Code and project needs to compile and run

- Make sure that your program compiles and runs
    - Warning level ALL …
    - NO Warnings or ERRORS
        - Your code should be squeaky clean.
    - Code needs to work "as-is".
        - No modifications to files or deleting files necessary to compile or run.
    - All your code must compile from perforce with no modifications.
        - Otherwise it's a 0, no exceptions

Leave Project Settings

- Do NOT change the project or warning level
    - Any changing of level or suppression of warnings is an integrity issue

Project needs to run to completion

- If it crashes for any reason…
    - It will not be graded and you get a 0

Simple C++

- No modern C++
  - No Lambdas, Autos, templates, etc…
  - No Boost
- NO Streams
  - Used fopen, fread, fwrite…
- No code in MACROS
  - Code needs to be in cpp files to see and debug it easy
- ***Exception:***
  - implicit problem needs templates

Leaking Memory

- If the program leaks memory
  - You will lose points
- If a class creates an object using new/malloc
  - It is responsible for its deletion
- Any ***MEMORY*** dynamically allocated that isn't freed up is ***LEAKING***
  - Leaking is ***HORRIBLE***, so you lose points

No Debug code or files disabled

- Make sure the program is returned to the original state
  - If you added debug code, please return to original state
- If you disabled file, you need to re-enable the files
  - All files must be active to get credit.
  - Better to lose points for unit tests than to disable and lose all points

Adding files to this project - NOT ALLOWED

- NO extra files

Use secured versions of string functions when needed

- Such as strcpy_s()

Due Dates

- Due 18 November Thursday at 5pm CST
  - Should take 2-3 hours but you can spend as much time as you want
  - Don't miss the due time.. or you fail the class!
- Submit program perforce in your student directory assignment supplied.
- Fill out your this ***Submission Report*** and commit to perforce
  - ***ONLY*** use Adobe Reader to fill out form, all others will be rejected.
  - Fill out the form and discussion for full credit.

Goals

- TAKE HOME Final Exam – sponsored by Jello – **There's Always Room for JELL-O**

Assignments

- ***Implement 5 coding problems***
    - DO NOT share answers or use social network to group work on final exam
        - All work is done by the individual
    - You can use the internet for references and look up only
    - DO NOT post any questions or support on Piazza or other sites
        - Only clarification if needed (should be self-explanatory)
    - There are no unit tests supplied
        - After all this is a final exam
    - DO NOT add files to the project.
    - DO NOT modify any project settings.
    - DO NOT leak memory

Problem 1: STL Sort

```
// ----------------------------------------------
// Sort the stl vectors:
//
// Sort by Median value (largest first)
//     If there is a tie.. use strict weak ordering
//     largest number is first and so on...
//
// Assume:
//     vOut is initally empty
//     vIn is read-only
//     Sorted array is stored in vOut
//
// Example:
//    vIn (input):
//        2 3 4 5 5 --> Median: 4
//        8 6 7 2 5 --> Median: 6
//        5 6 4 5 8 --> Median: 5
//        3 2 1 3 5 --> Median: 3
//        9 5 2 3 6 --> Median: 5
//        2 3 4 1 2 --> Median: 2
//        9 8 5 1 5 --> Median: 5
//
//    vOut (output):
//        8 6 7 2 5 --> Median: 6
//        9 8 5 1 5 --> Median: 5
//        9 5 2 3 6 --> Median: 5
//        5 6 4 5 8 --> Median: 5
//        2 3 4 5 5 --> Median: 4
//        3 2 1 3 5 --> Median: 3
//        2 3 4 1 2 --> Median: 2
//
// Hopefully you see the obvious pattern
//
// ----------------------------------------------

void SortMe(const std::vector< vData >& vIn, std::vector< vData >& vOut)
```

- Fill in the function and add any helper methods you want
- You also have access to the vData class
- Do not change or add data to the structure

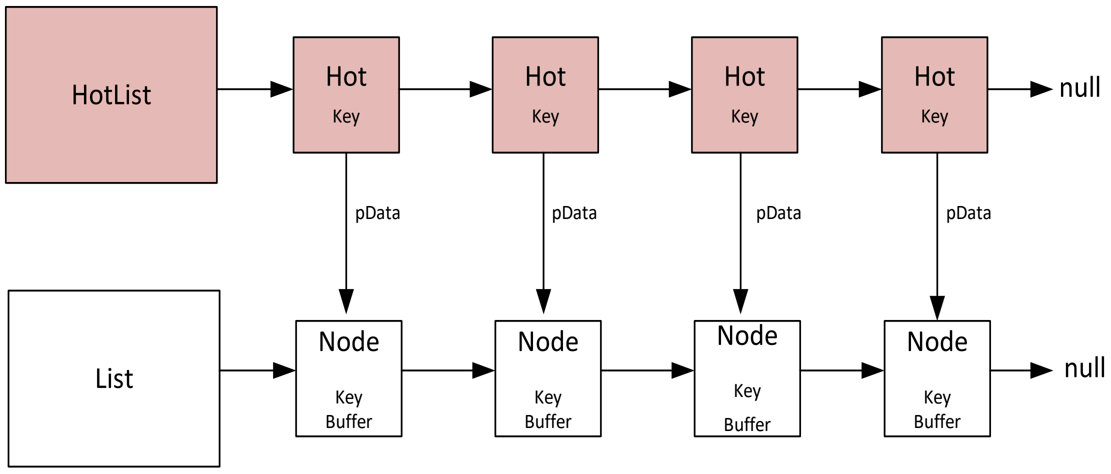
```
struct Vect
{
       int a;
       int b;
       int c;
       int d;
       int e;
}
```

- Do not ADD any files to the project

Problem 2: Cache optimized

- Do not ADD any files to the project
- Create a cache friendly single linked list called ***HotList***
  - Each ***Hot*** node holds the key and points to the original node
- The length of the original linked list is unknown
  - Its null terminated

```
class Hot
{
public:
        Hot() = default;
        Hot(const Hot &) = default;
        Hot &operator = (const Hot &) = default;
        ~Hot() = default;

        // -------------------------------------------
        // Data:  (do not add or modify the data)
        // -------------------------------------------

        Hot *pNext;
        unsigned int key;
        Node *pData;
};
```



- Create a Hot linked list, that is null terminated
  - Do NOT create all the Hot nodes in a Array or Memory Block or use placement new.
    - Instead create each Hot node individually with new.
    - You are creating many Hot nodes the corresponds to each Node in List.
  - You do not know the number of original nodes in the List…
    - Create the Hot nodes while walking(iterating) the original list.
- Create 3 functions:
  - Create a **constructor** to create the Hot List
  - Create the corresponding **destructor**
  - Create a **Find** function

- Using the HotList you can quickly find the node by searching for a specific key
    - Roughly 2-4x faster than the original even in Debug

// Sample Code

```
List *pList = new List();
HotList *pHotList = new HotList(pList);

PerformanceTimer t1;
PerformanceTimer t2;

// start timer
t1.Tic();
       Node *pTmp = pList->Find(0x36ca2b0e);
t1.Toc();

Trace::out("Key:%x time: %f ms \n", pTmp->key, t1.TimeInSeconds() * 1000);

// start timer
t2.Tic();
       Hot *pHotTmp = pHotList->Find(0x36ca2b0e);
t2.Toc();

Trace::out("Key:%x time: %f ms \n", pHotTmp->key, t2.TimeInSeconds() * 1000);
float ratio = t1.TimeInSeconds() / t2.TimeInSeconds();

Trace::out("Ratio: %f \n", ratio);

delete pList;
delete pHotList;
```

Problem 3: Proxy

Applying what you know. Refactor the Vect class to add a proxy to prevent unnecessary sqrt() calls.

- Len of a Vect
    - A.Len() = sqrtf(A.x*A.x + A.y*A.y + A.z*A.z);
- Comparing the length of two vectors can be done with the length squared
    - For example: If (A.Len() > B.Len())
        - Instead of calling the actual length…
            - You can compare the length squared of each vector.
    - Length squared
        - Length squared = A.Len() * A.Len();
        - Length squared = (A.x*A.x + A.y*A.y + A.z*A.z) ← no sqrt() much faster
- Add a proxy to Vect class to remove the need for sqrt() inside comparison operations
    - ==, !=, >, >=, <, <= comparison operators
        - If(A.Len() == B.Len())… ← no sqrt() calls
        - If(A.Len() != B.Len())… ← no sqrt() calls
        - If(A.Len() > B.Len())… ← no sqrt() calls
        - If(A.Len() >= B.Len())… ← no sqrt() calls
        - If(A.Len() < B.Len())… ← no sqrt() calls
        - If(A.Len() <= B.Len())… ← no sqrt() calls
    - The above 6 operators should not CALL sqrt() function when used with Len() …
        - compare with the squared length instead
- A solo method that return length is allowed to call sqrt()
    - Example: float val = A.Len(); ← this is allowed a sqrt()
- Use ***CDM::Sqrt()*** for *ALL* sqrt calls… Need to monitor the use of sqrt() in testing
- Do not ADD any files to the project

```
// -----------------------------------------------------------------------------------------------------------
// Please REFACTOR Vect class, feel free to add/delete/modify any method.
//      Add a Proxy structures/classes to accomplish the goal:
//          Len() method should _NOT_ call CDM::Sqrt() for comparison operators
//                      ==, !=, >, >=, <, <=    (no sqrt() calls allowed)
//          float val = A.Len();   ( is allowed to call CDM::Sqrt() )
//      You will need to change the existing code and refactor.
// -----------------------------------------------------------------------------------------------------------

class Vect
{
public:
	Vect() = default;
	Vect(const Vect &) = default;
	Vect &operator = (const Vect &) = default;
	~Vect() = default;

	Vect(float a, float b, float c);
```

```
        // Add or modify (or proxy) methods here:
        float Len();

// ------------------------------------------
// Data:  (do not add or modify the data)
// ------------------------------------------
private:
        float x;
        float y;
        float z;
};
```

// ---
// this is the sample test function, should work as is, leave it alone.
// ---

// Sample Code:

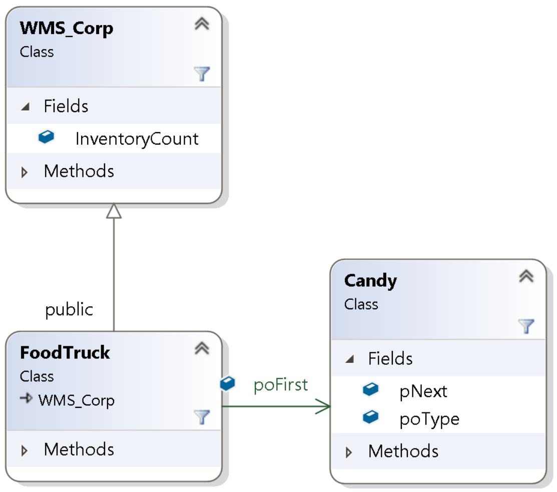
```
        Vect A(1, 2, 3);
        Vect B(3, 4, 5);
        float val1;
        float val2;

        val1 = A.Len();   // ← calls CDM::Sqrt()
        val2 = B.Len();   // ← calls CDM::Sqrt()

        if(B.Len() == A.Len())  // ← no sqrt() calls
        {
             Trace::out("1\n");
        }

        if(B.Len() != A.Len())  // ← no sqrt() calls
        {
             Trace::out("1\n");
        }

        if(B.Len() > A.Len())  // ← no sqrt() calls
        {
             Trace::out("1\n");
        }

        if(B.Len() >= A.Len())  // ← no sqrt() calls
        {
             Trace::out("1\n");
        }

        if(B.Len() < A.Len())  // ← no sqrt() calls
        {
             Trace::out("1\n");
        }

        if(B.Len() <= A.Len())  // ← no sqrt() calls
        {
             Trace::out("1\n");
        }
```

Problem 4: Memory Leak

- Rework the CLASSES to prevent MEMORY LEAKS
- Do not ADD any files to the project

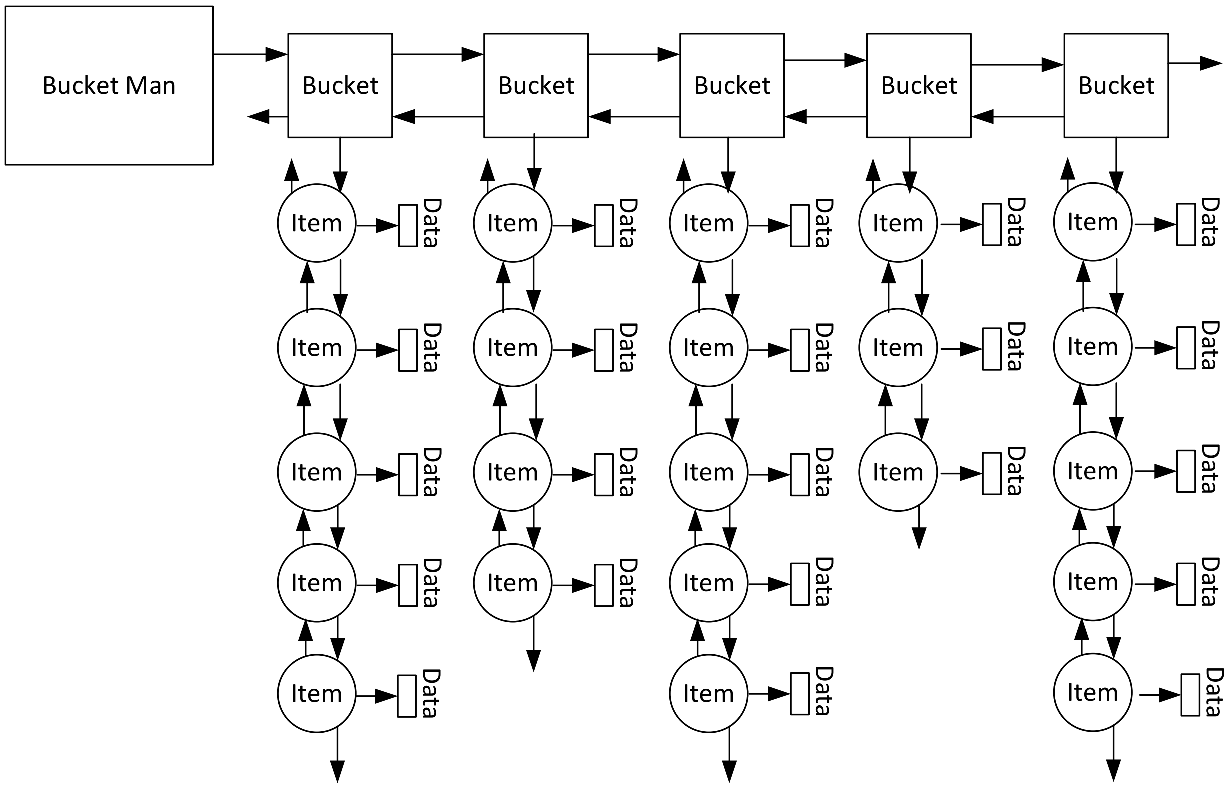
// --
// this is the sample test function, should work as is, leave it alone.
// Leave Print() method "as-is"
// --

```
int main()
{
    Candy *pA = new Candy("Pop Rocks");
    Candy *pB = new Candy("Bomb Pop");
    Candy *pC = new Candy("Nerds");
    Candy *pD = new Candy("KitKat");

    WMS_Corp *pWMS_Corp = new FoodTruck();

    pWMS_Corp->Add(pA);
    pWMS_Corp->Add(pB);
    pWMS_Corp->Add(pC);
    pWMS_Corp->Add(pD);

    pWMS_Corp->Print();

    delete pWMS_Corp;
}
```



Problem 5: Linked List

- Given a linked list structure.
    - ***Bucket Manager*** has multiple double linked ***Buckets***.
    - Each ***Bucket*** has double linked Items.
    - Write ***ONLY*** the ***destructors*** for all classes.
- Do not ADD any files to the project
- Assume that a complete environment is created for you.
    - You only write the destructors, nothing else!
    - I will show you a sample of typical environment
- Assume:
    - All objects are dynamically allocated with new
        - ***BucketMan, Buckets, Items and Data***
    - You can **delete** any one Item individually
    - You can **delete** any one bucket individually
    - You can **delete** the bucket manager
- Ownership
    - ***BucketMan*** – owns a group of Buckets
    - ***Buckets*** – owns its group of Items
    - ***Items*** - owns Data
    - Make sure all the appropriate items will be correctly deleted.
- I provided print function to make your life 1000x easier
    - Please use them



Sample creation environment

```
BucketMan *pMan = new BucketMan();

Bucket *pB3 = new Bucket(Bucket::name::B3);
Bucket *pB2 = new Bucket(Bucket::name::B2);
Bucket *pB1 = new Bucket(Bucket::name::B1);
Bucket *pB0 = new Bucket(Bucket::name::B0);

pMan->Add(pB3);
pMan->Add(pB2);
pMan->Add(pB1);
pMan->Add(pB0);

Item *p0 = new Item(Item::name::I3, Data::name::A);
Item *p1 = new Item(Item::name::I2, Data::name::B);
Item *p2 = new Item(Item::name::I1, Data::name::C);
Item *p3 = new Item(Item::name::I0, Data::name::D);

pB0->Add(p0);
pB0->Add(p1);
pB0->Add(p2);
pB0->Add(p3);

Item *p4 = new Item(Item::name::I1, Data::name::E);
Item *p5 = new Item(Item::name::I3, Data::name::F);
Item *p6 = new Item(Item::name::I4, Data::name::G);

pB1->Add(p4);
```

```
pB1->Add(p5);
pB1->Add(p6);

Item *p7 = new Item(Item::name::I5, Data::name::H);
Item *p8 = new Item(Item::name::I2, Data::name::I);
Item *p9 = new Item(Item::name::I7, Data::name::J);
Item *p10 = new Item(Item::name::I3, Data::name::K);
Item *p11 = new Item(Item::name::I6, Data::name::L);

pB2->Add(p7);
pB2->Add(p8);
pB2->Add(p9);
pB2->Add(p10);
pB2->Add(p11);

Item *p12 = new Item(Item::name::I4, Data::name::M);
Item *p13 = new Item(Item::name::I7, Data::name::N);
Item *p14 = new Item(Item::name::I1, Data::name::O);
Item *p15 = new Item(Item::name::I3, Data::name::P);
Item *p16 = new Item(Item::name::I0, Data::name::Q);

pB3->Add(p12);
pB3->Add(p13);
pB3->Add(p14);
pB3->Add(p15);
pB3->Add(p16);
```

Validation

Simple checklist to make sure that everything is submitted correctly

- Is the project compiling and running without any errors or warnings?
- Does the project run ***ALL*** the unit tests execute without crashing?
- Is the submission report filled in and submitted to perforce?
- Follow the verification process for perforce
  - Is all the code there and compiles “as-is”?
  - No extra files
- Is the project leaking memory?

Hints

- Good LUCK – You will do well!