Spencer’s answers are given in **red** below.

Sledgehammer Games Programming Test – Confidential

## Written (50 pts)

1. The following function occasionally crashes when called. What problems do you see that might explain the crash? Why might the crash only be occasional? (5 pts)

void CleanEntities(Entity \*entityHead)

{

Entity \*e = entityHead;

while(e)

{

if(e->bRemoveMe)

{

delete e;

}

e = e->next;

}

}

Answer:

The statement “e->next” may occasionally be a dereferencing of a stale pointer. Recall that “e->next” is another way of writing “(\*e).next”. Another reason for the occasional instability may be due to the freed memory still being valid. At other times, if another thread intervenes, it may reclaim the freed member, then when execution returns to this thread and this loop, it’s working with unintentionally modified data.

2. You are programming a sideways scrolling shooter. This uses 2D cell based maps that are very long (over 100,000 cells per row long, 24 rows of 40 cells on screen), and the only way you can get them to fit in memory is to compress each row of the map using run length encoding and to decompress each new column of the map on the right of the screen as the background moves from right to left. You maintain a list of pointers into the RLE data for each row of the screen. Columns that are scrolled off the left are discarded. This works very well and the game only just fit in the available memory.

Suddenly the game designer decides that the game should also be able to scroll from left to right at any point, and for any duration. Stunned and dismayed, you explain to him that RLE compression only works in one direction, but he’s adamant.

Figure out a way of doing what he wants without using much (no more that 1K of) extra memory. Explain the algorithm used for decompressing in both directions. (assume the RLE algorithm used is: a positive byte N followed by a byte B, indicates a run of N copies of B, a negative byte -N followed by N bytes indicates that those bytes should be copied directly. For example, the sequence 2,2,2,2,5,1,2,9,9,9,9,9 would be coded 4,2, -3,5,1,2, 5,9.) (15 pts)

Answer:

I don’t see why we’d really need to use any more memory to give the designer what she wants. At the start of the game, we’re scrolled all the way to the left -- we’re viewing the uncompressed region of the screen that is left-most, and the rest of the terrain is compressed to the right. Our uncompressed window just needs to be able to slide forwards and backwards along the compressed stream, and that really doesn’t significantly change the amount of memory we’re using up. The act of sliding the window left or right by one byte doesn’t cause our memory requirements to increase or decrease by any significant amount. (But it does change just a little bit, and I guess that’s where the 1K comes into play.) Anyhow, streaming-in the RLE from compressed data to decompressed data is already implemented. Streaming the RLE from decompressed data to compressed data should be just as easy to implement. Once you have those figure out, you can use them both at the same time while scrolling the decompression window left or right. There’s not much more to say than to perhaps talk about the streaming piece of it. When you concatenate a compressed stream with a raw bite, the compressed stream can easily absorb the byte into the RLE code. Similarly, if you want to pull one raw bite out of an RLE code, that’s easy to do as well. Determine the first byte in the RLE code, then fixup the RLE code to reflect the byte having been taken out. Hopefully all that made sense. Doesn’t sound hard to do.

3. How do the following function and macro definitions differ in behavior and side effects? How could the macro be improved? (5 pts)

#define max(a, b) a > b ? a : b

int max(int a, int b) { return a > b ? a : b; }

Answer:

The macro version will always be inlined while the function version may not unless the “inline” keyword is used. The macro, unlike the function, may suffer from operator precedence problems. To remedy this, each of “a” and “b” should be wrapped in parenthesis as “(a)” and “(b)”. The C-preprocessor may expand “a” or “b” into an expression with operations that mess up the intended order of operations. I would also wrap the first sub-expression of the ternary operator to be safe too.

4. You are working on a first person shooter and want to implement a feature where the player gets points if they shoot circular targets. The targets can be placed in the world at an arbitrary location with an arbitrary orientation and arbitrary size. Shots are very fast moving projectiles. Describe a data structure that would define a target, and write a routine to detect whether a particular target has been hit by a particular projectile. Assume the game follows a basic main loop architecture. Use C/C++ or pseudo code. (15 pts)

Answer:

I would do something like the following…

struct Target

{

Vector3 center; // Center location of target

Vector3 norm; // Unit-vector normal to the surface of the circular disk.

Float radius; // The radius of the disk.

};

struct Projectile

{

Vector3 fireLocation; // The point at which the projectile was fired.

Vector3 fireDirection; // Unit-vector in direction the projectile was fired.

};

// Tell us if the given target is hit by the given projectile. This is just a ray-cast.

bool TargetHitByProjectile( const Target& target, const Projectile& projectile )

{

float dotDenom = VectorDot( projectile.fireDirection, target.norm );

if( dotDenom == 0.0f )

return false; // Direction of fire parallel to plane of target.

float dotNumer = VectorDot( target.center – projectile.fireLocation, target.norm );

float param = dotNumer / dotDenom;

if( param < 0.0f )

return false; // Projectile fired away from target.

Vector3 impactLocation = projectile.fireLocation + param \* projectile.fireDirection;

float distance = ( impactLocation – target.center ).Length();

if( distance > target.radius )

return false; // Ray missed target.

// We hit the target!

return true;

}

I did the algebra for the above subroutine on paper. Hopefully I didn’t make mistake. If I did, I could easily figure out where I went wrong while debugging the routine.

Now, this assumes that the bullets travel infinitely fast. If our projectiles traveled at some finite speed, then we might do something where we track the state of all projectiles and integrate them each frame, and check their collision each frame with potential targets.

5. Given the following String class and its defined operations, implement the destructor and the "+=" operator as declared below. Note the memory management assumptions implied by the implementation of the default constructor. (5 pts)

class String

{

public:

String() { m\_str = new char[1]; \*m\_str = 0; }

~String();

String& operator+= (const String& other);

private:

char\* m\_str;

};

Answer:

~String() { delete[] m\_str; }

String& operator+=( const String& other )

{

// I would do something along these lines, I suppose.

int curLen = strlen( m\_str );

int otherLen = strlen( other.m\_str );

int newLen = curLen + otherLen + 1;

char\* newStr = new char[ newLen ];

strcpy\_s( newStr, newLen, m\_str );

strcat\_s( newStr, newLen, other.m\_str );

delete[] m\_str;

m\_str = newStr;

}

If someone decided to create a new class that inherited from the above class in order to customize memory management for a certain category of game String, what issues might arise? Would there be any recommended changes to the Original String class above? (5 pts)

Answer:

It would be important to make the destructor of the base class virtual. Failing to do so can lead to memory leaks. You might put “m\_str” in a protected section, instead of private. Otherwise, the derived class would have to go through the base for any access or management of the buffer.

## Programming (50 pts)

## 

Please send back a working driver.cpp. The code must be production quality and compile error and warning (level 3) free under visual studio. You can use C or C++, **no STL.** (50pts) **Efficiency will be graded.**

1. Write a simple program (must compile) that take as input a board of NxM square (1x1) cells where each cell may have a non-uniform height of x.

Write a program that outputs the total volume of standing water the board can hold after submerged underwater and lifted back out. You can assume when lifting the board out no water is spilled.

*For example given a board where each number below represents the cell height*

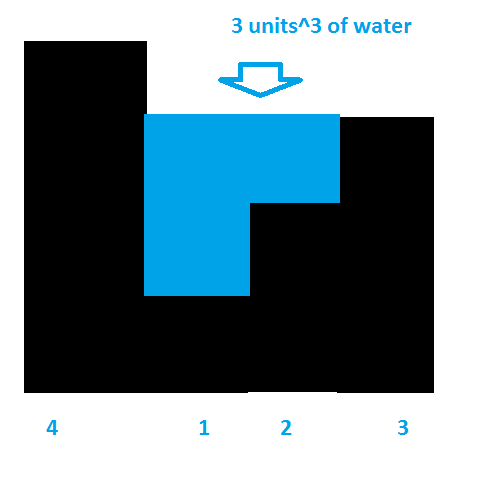
*4 3 3*

*4 2 3*

*4 1 4*

*4 4 4*

Looking at a cross section of the middle column



*The volume returned for this board is: 3 units cubed*

*4 3 3*

*4 2 3*

*4 1 1*

*4 4 4*

*The volume returned for this board is: 0 units cubed*

What is the running time of your algorithm?

The algorithm I’ve given in **driver.cpp** (attached), I believe, is O(N^3). I’m not sure of a more efficient way to solve the problem, assuming my solution is correct in the first place.

Your function MUST follow this signature:

//@param int const \* const unsigned short \*a\_board - argument representing the NxM board.

//@param unsigned short a\_w - argument representing the width of the board

//@param unsigned short a\_h - argument representing the height of the board

//@return - unsigned \_\_int64 - the volume of water the board retains

unsigned \_\_int64 CalculateVolume( const unsigned short \*a\_board, unsigned short a\_w, unsigned short a\_h )

**See the attached file “driver.cpp”**