**1) In the following C++ code what is the value of v?**

unsigned int v = 1 << 10;

* 2 raised to 10 = 1024

**2) What are the minimum and maximum values that can be stored in a signed char?**

**- -128 to +127**

**3) In the following C++ code what is the value of v?** (in c++ ^ is the XOR operator)

unsigned int v = (129^130);

Ans = 2 – incorrect. Answer is 3.

**4) Convert 0xFE12ED78 from big-endian to little-endian.**

**0x78ED12FE**

**5a) What can a quaternion be used to represent?**

* **Rotation**

**b) Why would you use a quaternion as opposed to a matrix?**

**- Smaller memory requirement. SLERP.**

**6a) What is the dotproduct of** (0.5f, 2.0f, 0.0f) and (-4.0f, 1.0f, 7.0f).

**= 0.0f**

**b) What does the answer tell us about the relationship between these 2 vectors?**

**-Perpendicular to each other**

**c) What practical implications does the dot product have in AI? Can you think of an example?**

**-Finding angle between 2 characters or the AI and a point in the world (any object). Can be used for Line-of-Sight checks, Melee attach targeting.**

**7a) What is the result of the cross product of two vectors?**

**-Vector perpendicular to both vectors in question.**

**b) If the result has a magnitude of zero, what does that tell us?**

**-Vectors are parallel to each other or the same vector cross product with itself.**

**8) Why would you use a linked list vs a vector?**

**-Insertion has very low memory footprint – no copying, allocation can be anywhere. Inserting in the middle is easier but still requires finding the right location.**

**- Deletion has little overhead. Can be not an advantage if vector uses swap with back and delete trick.**

**9a) Can you describe what a hash map is and why you might use it?**

**-A key-value pair structure. Key is hashed with a hashing function to find place to insert. Key is the index to find value.**

**-Can be used to implement a dictionary of not just words.**

**b) What is the best case access time for a hash map?**

**-Constant – O(1) . No collisions**

**10) You are asked to keep track of the last 5 characters that attacked the player. How would you store this information?**

**-An array with fixed size that is a member of an actor component on the player character. Handle of the attacking characters will be stored – or weak pointer to them.**

**11a) What is the A\* algorithm, and what are its advantages over similar algorithms?**

**-A path finding algorithm.**

**-Advantage is it uses a heuristic function to determine if it is getting closer to the destination. Other algorithms only look at the path length in the graph. Others can go in the wrong direction because of this.**

**b) What data structures may be involved in a practical implementation of A\*?**

**-A queue to keep track of nodes. A class for each node that stores path length, heuristic data.**

**12) Briefly describe what an FSM is.**

**-Finite State Machine. Defines states that, example, a character can be in. You can only be in 1 state at a time. There are transition edges from sate to state.**

**13) Explain hysteresis and give an AI related example.**

**-AI won’t switch from 1 task to the new one immediately. Helpful to prevent constant and untimely switching between tasks.**

**14) Re-write the following function so that it is more computationally efficient**

bool IsHypotenuseLessThan10( float fEdge1, float fEdge2 )

{

float fHypotenuse = (fEdge1\*fEdge1) + (fEdge2\*fEdge2);

return (sqrt(fHypotenuse) < 10.0f);

}

-Don’t do sqrt. Compare with square of 10.

**15) At a high level, what kinds of things could you do to optimise a scene with thousands of fully physical, moving entities.**

**-Spatial partitioning, BSTs.**

**-LOD**

**-Don’t evaluate some things for entities that are really far.**

**-Collisions can be checked only based on partitions.**

**16) What is a race condition? What mechanisms can you use to prevent a race condition from occurring?**

**-When the same data is accessed in conflicting way by different functions.**

**-To prevent Use mutex, lock etc. Better is atomic lock.**

**17) What is the construction order of an inherited class in C++?**

**-Base class first, then child class. For each class, members are initialized in order of declaration in class.**

**18) Why might you use dynamic\_cast here?**

void Draw( CEntity\* pEntity )

{

ASSERT(dynamic\_cast<CDrawable\*>(pEntity));

CDrawable\* pDrawable = static\_cast<CDrawable\*>(pEntity);

pDrawable->Draw();

}

-To ensure pDrawable is pointing to a complete object.

**19) What is the output for this function?**

#include <stdio.h>

#include <stdlib.h>

int main(int argc, char\* argv[])

{

      int v = 4;

      int value1 = ++v;

      int value2 = v++;

      printf("value1 = %d\n", value1);

      printf("value2 = %d\n", value2);

      return 0;

}

value1 = 5

value2 = 5

**20) What’s wrong with this function?**

float GetStrengthAtDist(float dist, float strength, float r1, float r2)

{

float result = strength;

if(dist > r2)

{

result = 0.0f;

}

else if(dist > r1)

{

float result = r2 – dist;

result /= r2 – r1;

result \*= strength;

}

return result;

}

- r2 – r1 could be 0 – divide by zero error. Use assert.

**21) The following classes describe the visualisation of 2 shapes; a square and a circle.**

class CSquare

{

public:

CSquare(float fSize, Vector centre) {...}

void Draw() {...}

void GetBoundingBox(Vector& vMin, Vector& vMax) {...}

Colour& GetColour() { return m\_colour; )

private:

Colour m\_colour;

float m\_size;

Vector m\_centre;

};

class CCircle

{

public:

CCircle(float fRadius, Vector centre) {...}

void GetBoundingBox(Vector& vMin, Vector& vMax) {...}

void Draw() {...}

Colour& GetColour() { return m\_colour; )

private:

Colour m\_colour;

float m\_radius;

Vector m\_centre;

};

1. **Using inheritance Re-write the 2 classes above by adding in a new class CShape that they derive from, moving as much of the common functionality into the new class as is desirable.**

**-Can discuss. Out of time.**

1. **Why is it useful for the 2 shapes to derive from a common base class?**

**- Can discuss. Out of time.**

1. **Describe what a pure virtual function is and why it is useful in your implementation of CShape.**

**- Can discuss. Out of time.**

**22) What’s possible issues can you foresee with this class? Note: It does compile!**

class CBiped

{

public:

CBiped();

~CBiped();

void CreateGun();

CGun\* GetGun() const {return m\_pGun;} // should return a const CGun\*

void SetGun(CGun\* pGun);

private:

CGun\* m\_pGun;

};

CBiped::CBiped() : m\_pGun(NULL)

{

}

CBiped::~CBiped()

{

}

void CBiped::CreateGun()

{

m\_pGun = new CGun(); //not freed in dtor

}

void CBiped::SetGun(CGun\* pGun)

{

m\_pGun = pGun; //not freeing held gun memory. Could become dangling ptr when in\_param is deleted.

}