|  |  |
| --- | --- |
| Name: |  |
| Date: |  |
| Position Sought: |  |

**PROGRAMMER TEST**

This test is intended to help us evaluate your basic programming skills, your general knowledge and understanding of programming related issues, to see how you approach certain type of problems, and to get a broad idea of your written communication skills.

The test takes a couple of hours to complete. Please try to answer all questions as fully as you think would be necessary to convey to us that you understand both the problem and the solution. Providing several alternative solutions, where possible, is encouraged. At the same time, try to be concise and to the point—avoid writing essays.

Please repeat the questions along with your answers. If you find a problem specification lacking, feel free to point this out along with any assumptions you’ve made.

If you think a question is stupid, please state so, and why you think that is the case. Also, if you’re not sure how to go about solving a problem, it is OK to say so (but if you have some hunches, feel free to list them).

**PROGRAMMING PRACTICES**

1)    Briefly discuss the advantages and disadvantages of high-level vs. low-level languages. When would you use the one over the other?

2)    Similarly, what do you consider to be the advantages and disadvantages of C vs. C++?

3)    What considerations are important when working on a team?

4)    Culling is a concept that frequently crops up in games and other real-time (graphics) applications. On a concept level briefly describe as many different forms of culling as you can think of; how they work, and where they would be applicable in a game.

5)    In a paragraph or two each, explain the following three topics and describe game programming applications for them: fixed-point arithmetic, dot and cross product, hash tables.

**PROBLEM SOLVING**

6)    You are repeatedly polling a hardware register containing a single bit of interest, resulting in a binary data stream of input values. This data stream needs to be filtered in such a way that in all sequences of consecutive one bits, only the first one bit is left at one, and all other bits in that same sequence are set to zero. For example, it would change the data stream 010011110110 to 010010000100. What would your code for this look like? If there were additional bits in the same register providing other bit streams, how would you change your code to handle this?

7)   You are working on a 3D platform game, where the main character needs to jump, the jumps being affected by gravity. How do you implement this? How would you add a variable-height jump feature, where the character jumps higher when the jump button is pressed for a longer period of time? Finally, how would you implement a double jump, allowing the player to get even higher by pressing the jump button again near the apex of the jump?

8)    In a real-time game, a computer-controlled character is roaming an open two-dimensional MxN gridded area, moving from grid square to grid square. Within the area, there are a few large irregular obstructions covering a number of grid squares each. Initially the character is moving straight towards the player when it encounters an obstacle. How can it decide whether it is better to go left or right to faster make it to the player? How could you enhance the map (by storing extra information) to speed up this decision making?

9)    You have a 3D racing game featuring a track of varying width that in several places branches one or more times, loops around in wild ways, paths sometimes merging back into each other before finally turning into a smooth finishing straight, where the goal and start line is one and the same. How would you implementing a leaderboard feature that at all times correctly displays who is in first place, second place, etc, up to final place during, say, a three-lap race?

10)   How would you implement (and represent) the player-world collision detection in a snowboarding game where you are sometimes on top of hard-packed snow, sometimes knee-deep in powder; where you can slide down railings as well as crash into trees and building walls? In addition, how would you handle surface modifiers, e.g. force-feedback when on the gravel patch, within your system?

11) You are called upon to implement an AI bot to play a game of Quake IV as well as or better than the best of human players. Identify what the tricky (as well as non-tricky) issues are and outline possible approaches to addressing those

12) Without use of other function calls, rewrite this code to ensure the pointer p is aligned to a 16-byte boundary for its lifetime:  
     
   char \*p = malloc(1024);  
   ...  
   free(p);

13)  Flat on a plane (with normal N through point P) sits a tank at point Q. The tank's local coordinate system is described by the 3x3 rotation matrix M. How do you update M and Q to make the tank first rotate in-place 45 degrees to its right, followed by moving K units forward? Please express your solution in math, not text. Also explain any assumptions made.

14) Given the matrix equation A \* B \* C = D, assume A, C, and D, are given, and that B can be solved for. Show the steps needed to obtain B.

15) Given two axis-aligned bounding boxes (AABBs), give code for determining if \_either\_ box lies fully inside the other.  
Strive to make your solution efficient (and explain what efficient means in terms of your solution and for a modern CPU).

**DEBUGGING**

16)    Your program is working fine, when you make a few modifications to the code and it suddenly crashes. You undo the changes you made, recompile, but the code still crashes! What explanations can you come up with for this scenario happening? Which is the most plausible one?

17)    The following code contains a few bugs and potential pitfalls. Try to identify them.

I’ve inserted my comments right into the original code:

1: #define N 100;

2: unsigned int buf[N];

3:

4: // this function rotates the buffer left or right depending on opcode value in c

5: int weird(char c, int n, int high, int low, void \*p)

6: {

7:     unsigned int \*pmiddle, i;

8:

9:     if (c & 0x0f) {

10:      // erase the +-n middle values of buf to high nibble

11:      pmiddle = &buf[N/2];

12:      for (i = -n; i <= n; i++)

13:         pmiddle[i] = c >> 4;

14: } else if (c == 0xf0) {

15:      // initialize buffer to high\*2^16+low

16:      for (i = 0; i <= N; i++)

17:          buf[i] = high << 16 + low;

18: } else if (c == 0xf1) {

19:      // set bit 31 of all words in buffer to disable

20:      for (i = 0; i < n; i++) ;

21:          buf[i] += 1<<31;

22: } else {

23:      // illegal opcode

24:      printf("opcode %s has buf value=%d/n", c, pmiddle[c]);

25:      break;

26: }

27: }