PA9 – Programming Assessment

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: 11/13/2021

Submission Details

Final ***Changelist*** number: 123399

Verified build: (•) Yes ( ) No

Number Tests Passed: DNA

Required Configurations: Just CPP files

Discussion (What did you learn):

I learned:
- that using enum's as return type's is pretty coo
- That making code robust is a pain in the ass sometimes.
- That I get as worked up about a mock code test as much as a real one.
- That I have no clue how to programmatically calculat a linear regression formula.

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

Submit multiple times to Perforce

- Submit your work as you go to perforce several times (at least 5)
  - As soon as you get something working, submit to perforce
  - Have reasonable check-in comments
    - Points will be deducted if minimum is not reached

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

Project needs to run to completion

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

No Containers

- NO STL allowed {Vector, Lists, Sets, etc...}
  - No automatic containers or arrays
  - You need to do this the old fashion way - ***YOU EARNED IT***

Leave Project Settings

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

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
  - There is a deduction of 20% of grade
- 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

No Adding files to this project

- This project will work “as-is” do not add files…
- Grading system will overwrite project settings and will ignore any student’s added files and will returned program to the original state

UnitTestConfiguration file (if provided) needs to be set by user

- Grading will be on the UnitTestConfiguration settings
  - Please explicitly set which tests you want graded… no regrading if set incorrectly

Due Dates

- See Piazza for due date and time
- 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

- Programming Assessment
    - Real world C++ exam that we give to interview candidates
    - Great practice
- ***DO NOT*** talk about ***ASSESSMENT*** on ***PIAZZA***
    - This assignment needs to be private – no discussions

Assignments

- Write all programs in cross-platform C or C++.
    - ***Optimize for execution speed and robustness.***

- Create a programming file for each problem, for example
    - Student directory - for this assignment ***only the CPP files submitted***
        - /PA9/problem1.cpp
        - /PA9/problem2.cpp
        - /PA9/problem3.cpp
        - /PA9/problem4.cpp
    - NOTHING else.. no project or other code or material

- Do all your work by yourself
    - Feel free to talk with others about setup, version control
    - ***DO NOT SHARE*** coding ideas on this assignment
    - Do not copy your friend's code.
        - This is a competition!

- NO SIMD
    - Straight C++

- There is a sample problem4.cpp in that directory that you can use as a reference to refactor. (saves typing)

- Read the instructions carefully
  - Optimize all code for ***Speed*** and ***Robustness***
  - If your program does not compile, it is a zero.
    - Only submit the existing cpp files, nothing else
  - Most students do ***VERY*** poorly on this assignment
    - They are rushing and not thinking about this assignment correctly
  - In class we go through all the optimizing ideas that you would need to successfully complete this assignment, please review

- C++ fundamentals sometimes trip up students
  - Think about the interfaces
    - Clean simple robust interfaces are needed
    - Think about the end user, how would they call the functions
  - Should you use pointers, references, or value?
  - What external functions are you calling?
    - Look up each function, make sure you understand what each function does
    - Think about how external functions are used from a performance point of view.
    - Can those routines give you errors?

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

Most assignments will have hints in a section like this.

- Do many little check-ins
  - Iteration is easy and it helps.
  - Perforce is good at it.

- ***DO NOT*** Use the FORUMs
  - This assignment is about your individual assessment
  - See me during office hours.
  - Read, explore, ask questions in class

Problems

Write all programs in cross-platform C++.

- Optimize for execution speed and robustness.

1) ***(C++)*** Write a function to calculate this equation:

$$y = a_0 + a_1x + a_2x^2 + a_3x^3 + a_4x^4 + a_5x^5$$

Assume: input: float x, output: float y
a0, a1, a2, a3, a4, a5 – are unique, non-zero numbers that do not change ever.

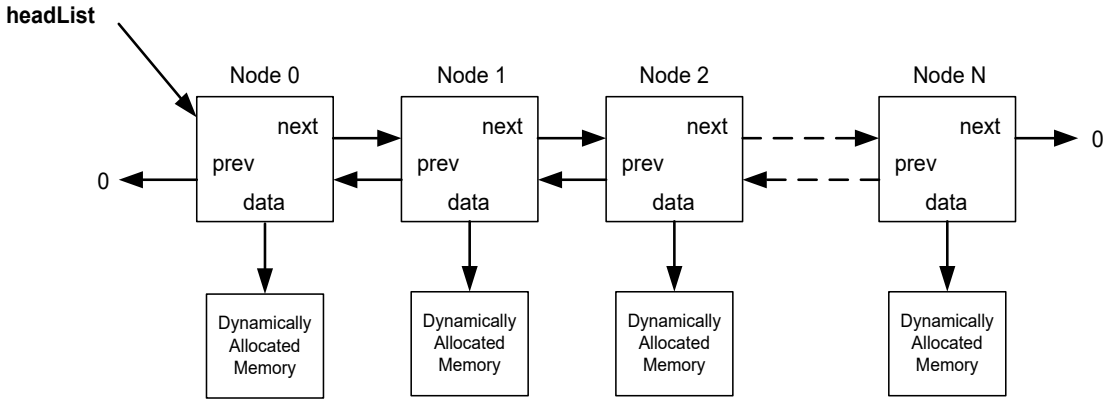
2) ***(C++)*** Write a stand-alone function (not a class/method) to create a unit vector from a 3D vector. Write the function and a simple structure for the Vector (no methods).

3) ***(C++)*** Write a function that removes ***a given node*** in a linked list container containing nodes of type LinkList. Assume that each LinkList node and its data pointer were originally allocated dynamically by new (C++ code) and is valid. No globals, all parameters need to be passed to the function. (*Pass the headList and the node as input arguments)*

```
class LinkList
{
public:
        LinkList        *next;
        LinkList        *prev;
        Data            *data;

};

LinkList *headList;
```



4) ***(C++)*** The findMaxDistance() function (given below) calculates the maximum distance between any two players in the passed-in player array. Refactor it so that it ***also*** calculates the minimum distance between any two players. You may change the signature of the function and the contents of the function, but do not change the Vect_t structure.

```
struct Vect  // Vector struct for positions
{
   float x;
   float y;
   float z;
};

/*************************************************************************
*
*  Function: findMaxDistance()
*
*  Input:
*          int       nPlayers - number of players
*          Vect      *playerArray - the array of players
*  Output:
*          float     maxDist - the maxDistance between any two players
*
*************************************************************************/

float findMaxDistance( int nPlayers, Vect *playerArray )
{
     int i,j;          // counter variables
     Vect_t tmpVect;   // temporary vector
     float tmpDist;    // temporary distance
     float maxDist;    // current max distance

     // initialize the distance to zero
     maxDist = 0.0f;

     for( i = 0; i < nPlayers; i++ )
        {
        for( j = 0; j < nPlayers; j++ )
           {
           // Find a vector between point i and j
           tmpVect.x = playerArray[i].x - playerArray [j].x;
           tmpVect.y = playerArray[i].y - playerArray [j].y;
           tmpVect.z = playerArray[i].z - playerArray [j].z;

           // Get its length
           tmpDist = (float)sqrt( tmpVect.x * tmpVect.x
                                + tmpVect.y * tmpVect.y
                                + tmpVect.z * tmpVect.z );

           // determine if it's a new maximum length
           if( tmpDist > maxDist)
              {
              // yes so keep it
              maxDist = tmpDist;
              }

           } //for(j)
        } // for(i)

    return maxDist;

} // End of findMaxDistance()
```