
DC
CROUSER
The art of Dean Crouser
www.crouserart.com
Blue Jay 2

Jolly-good Jay

In this quest, you get to write three small and relatively simple programs. My recommendation is to keep them as simple as possible. These functions ought not be longer than about 5-10 lines of code each. While you can get over-engineered programs to pass the test suite, they will be frowned upon in pro-circles and won't gain you an entry into prestigious quests.

As usual, I will give you template code you can copy and flesh out.

Your first miniquest - Schrodinger's cat

In the file called Draw_Cat.cpp, complete the function with the following signature:

```
void draw_cat();
```

When invoked, this function must print the following lines on the console. You may use one or more hard-coded "std::cout << . . . " statements to get the desired output.

```
  -------
 | /\_/\ |
 |( o o )|
 | > ^ < |
  -------
Schrodinger
```

The output must match exactly, character for character. Note the exact column spacings. There is no space after the last visible character on each line.

The `void` return type means that the function does not return a value back to its caller (in this case, your `main()` function).

Invoke this function from your main(). Thus when I run your program, it should print the cat on my console.

That's all.

Read this important note now

In all miniquests that require arithmetic calculations, you will find that there are multiple ways to calculate something.

For example, x = (a + b)/2.0 can be calculated as x = a/2.0 + b/2.0

But did you know that they are not the same in floating point arithmetic? You can't assume that (a+b)/2 will be exactly equal to a/2 + b/2. Why? Discuss it in the forums.

This has implications for your miniquests. Sometimes you will find that your results are very slightly different from mine. This means you're calculating the result a different way. You're not wrong to do so, but the challenge in the miniquest is to figure out (through intelligent trial-and-error) the exact calculation I'm using (without looking at my code or asking me) and replicate it at your end.

Avoiding the use of math library functions (except where noted as permitted) is a good start.

Your second miniquest - Limerick

Consider the following poem:

> *A dozen, a gross and a score*
> *and three times the square root of four*
> *divided by seven*
> *plus five times eleven*
> *is nine squared and not a bit more.*

Suppose I don't know how much a dozen, a gross or a score is. I'm going to use a function you create to discover values for these variables such that the above poem is true.

In a file called Limerick.cpp, flesh out the function:

```
double eval_limerick(int dozen, int gross, int score);
```

It should calculate the left-hand-side (LHS) of the above poem using the supplied values for dozen, gross and score and return the result. The LHS is all the terms before the "is". Thus your function should return the quantitative result of the calculation:

```
dozen + gross + score + 3 * sqrt(4) / 7 + 5 * 11
```

But before you jump the gun, make sure to calculate it by hand and confirm that you correctly get the RHS (81). Use parentheses in the above expression to make sure you get the operand groupings that make the poem come true. You are allowed to use the math library function, `sqrt()`.

Note: Yes. I know that you can implement the function using non-intuitive values for these quantities and still pass the test. Just use common sense values. It's more fun that way.

Your third miniquest - Etox

In a file called Etox.cpp, implement the following function:

```
double etox_5_terms(double x);
```

It should return the value of e^x back to its caller. This value should be calculated as the sum of exactly the first five terms in its expansion below:

$$e^x = 1 + x + x^2/2! + x^3/3! + . . .$$

where `n!` is the factorial of `n`, which is the product of all positive integers at most equal to `n`.

You would therefore return the result of the calculation:

$$1 + x + x^2/2! + x^3/3! + x^4/4!$$

Keep it simple and don't overcode. No need to write a function for calculating factorials. Simply use numeric literals in their place.

In your main() function, prompt the user as follows:

```
Enter a value for x:
```

There must be one space after the colon and no newline. You must accept console input on the same line as the prompt.

Now read the user's response into a variable, calculate e^x using the function you first defined, print the value on the console followed by a single newline.

How well you do in this quest depends a lot on your ability to pay attention to the small details of what is being asked. That's all. As I said, this week's quest is relatively easy. Hopefully you can solve it quickly and move on to the next one. But don't be in a hurry to face the main boss yet.

Starter code

Your `Draw_Cat.cpp`:

```
// Student ID: 12345678
// TODO - Replace the number above with your actual Student ID
//
//  Draw_Cat.cpp
//
//  When this program is run it should print the following lines on the console:
//  For reference, Schrodinger on the last output line starts at col 1.
//
//    -------
//   | /\_/\ |
//   |( o o )|
//   | > ^ < |
//    -------
//  Schrodinger
//
#include <iostream>
```

```
using namespace std;

void draw_cat() {
    // TODO - Your code here
}

int main(int argc, const char * argv[]) {

    // TODO - Invoke your function from here.
    // I'll invoke your main()

    return 0;
}
```

Your `Limerick.cpp`:

```
// Student ID: 12345678
// TODO - Replace the number above with your actual Student ID
//
//  Limerick.cpp
//  2a-Lab-01
//
#include <iostream>
#include <sstream> // Need this for istringstream below

#include <cmath>   // needed for sqrt
#include <cstdlib> // for exit()

using namespace std;

double eval_limerick(int dozen, int gross, int score) {
    // TODO - Your code here
}

// I'm using command line arguments below to let me test your program with
// various values from a batch file. You don't have to know the details for
// CS2A, but you're welcome to - Discuss in the forums any aspect of this
// program you don't understand.

int main(int argc, char **argv) {
    int dozen, gross, score;

    if (argc < 4) {
        cerr <<"Usage: limerick dozen-val gross-val score-val\n";
        exit(1);
    }
    istringstream(argv[1]) >>dozen;
    istringstream(argv[2]) >>gross;
    istringstream(argv[3]) >>score;

    // Invoke the eval_limerick function correctly and print the result
    // with a single newline at the end of the line.
    // TODO - Your code here (just invoke your function with the above
    // values for its params. Don't worry about argc, etc. for now)

    return 0;
}
```

As you can see, you don't have to make major modifications to the `main()` function for `Limerick`. But you do have to invoke your `eval_limerick function` using the values in your dozen, gross and score variables, and print the result.

Here is your `Etox.cpp` starter file:

```
// Student ID: 12345678
// TODO - Replace the number above with your actual Student ID
//  Etox.cpp
//  2a-Lab-01
//
#include <iostream>
#include <sstream>

#include <cmath>   // needed for sqrt
#include <cstdlib> // for exit()

using namespace std;

double etox_5_terms(double x) {
    // TODO - Your code here
}

int main(int argc, char **argv) {
    string user_input;
    double x;

    cout <<"Enter a value for x: ";
    getline(cin, user_input);
    istringstream(user_input) >>x;

    // TODO - Your code here

    return 0;
}
```

Discuss anything you don't understand in the public discussion forums on Canvas to get all the clarifications you need.

Testing your own code

You should test your functions using your own `main()` function in which you try and call your functions in many different ways and cross-check their return value against your hand-computed results. But when you submit you must NOT change the behavior of the main() method provided in the starter code above.

Submission

When you think you're happy with your code and it passes all your own tests, it is time to see if it will also pass mine.

1. Head over to https://quests.nonlinearmedia.org
2. Enter the secret password for this quest in the box
3. Drag and drop your three cpp files (Draw_Cat.cpp, Limerick.cpp and Etox.cpp) into the button and press it.
4. Wait for me to complete my tests and report back (usually a minute or less).

Points and Extra Credit Opportunities

I monitor the discussion forums closely and award extra credit points for well-thought out and helpful discussions.

May the best coders win. That may just be all of you.

Happy Hacking,

&