```
Script started on 2023-12-15 10:59:00-06:00 [TERM="xterm" TTY="/dev/pts/5" COLUMNS=
mf98604@ares:~$ pwd
/home/students/mf98604
mf98604@ares:~$ show-code distance.cpp
```

distance.cpp:

```cpp
#include <iostream>
#include <cmath>

using namespace std;

double get_distance(double first_x, double first_y, double second_x, double
    return sqrt(pow(second_x - first_x, 2) +
                pow(second_y - first_y, 2) * 1.0);
}

int main() {
    cout << "\n\t\tWelcome to the 2D Distance Program!!!\n";

    double first_x,
           first_y,
           second_x,
           second_y,
           distance;

    char symbol;

    cout << "\nEnter first point x and y coordinates (x, y): ";
    cin >> symbol >> first_x >> symbol >> first_y >> symbol;
    cout << "\nEnter second point x and y coordinates (x, y): ";
    cin >> symbol >> second_x >> symbol >> second_y >> symbol;

    cout << "\n\nThank you!!  Calculating...  ";
    distance = get_distance(first_x, first_y, second_x, second_y);
    cout << "Done." << "\n\n(" << first_x << ", " << first_y << ") is " <<
    distance << " units away from (" << second_x << ", " << second_y << ")

    cout << "\n\nThank you for using the TDP!!\n";
    cout << "\nHave a wonderful day!\n";
    return 0;
}
```

```
mf98604@ares:~$ show-code midpoint.cpp
```

midpoint.cpp:

```cpp
#include <iostream>

using namespace std;

int main()
{
    char symbol;

    double x1,
           y1,
           x2,
           y2;

    double midpoint_x,
           midpoint_y;

    cout << "\n\t\tWelcome to the 2D Midpoint Program!\n\n";

    cout << "What is the first endpoint? ";

    cin >> symbol >> x1 >> symbol >> y1 >> symbol;

    cout << "What is the second endpoint? ";

    cin >> symbol >> x2 >> symbol >> y2 >> symbol;

    cout << "\nCalculating... ";

    midpoint_x = ((x1 + x2) / 2);
    midpoint_y = ((y1 + y2) / 2);

    cout << "Done.\n";

    cout << "\nThe midpoint of the line segment between points "
            "(" << x1 << ", " << y1 << ") and (" << x2 << ", " << y2 << ")"
            " is (" << midpoint_x << ", " << midpoint_y << ").\n";

    cout << "\nThank you for using the 2D Midpoint Program!\n";

    cout << "\nHave a bright day!\n\n";

    return 0;
}
```

```
mf98604@ares:~$ show-code ounces.cpp
```

ounces.cpp:

```cpp
#include <iostream>
#include <limits>

using namespace std;

constexpr streamsize INF_FLAG{numeric_limits<streamsize>::max()};

int main()
{
```

```
    const short OZ_IN_LB = 16;

    short ounces, pounds, ounces_remaining;

    cout << "\n\t\tWelcome to the Ounce-to-Pound Conversion Program!!!\n\n'
    cout << "Enter number of ounces: ";

    cin >> ounces;
    cin.ignore(INF_FLAG, '\n');

    pounds = ounces / OZ_IN_LB;
    ounces_remaining = ounces % OZ_IN_LB;

    cout << '\n' << ounces << " oz. is equal to " << pounds << " lb(s) and
         << ounces_remaining << " oz. ("
         << static_cast<double>(ounces) / OZ_IN_LB << " lbs.).\n";

    cout << "\nThank you for using the OTP Conversion Program!!\n"
            "\nHave a wonderful day!\n\n";

    return 0;
}
```

mf98604@ares:~$ show-code time.cpp

time.cpp:

```
#include <iostream>
#include <string>
#include <ctime>

using namespace std;

bool is_leap(short year);

int main() {
    const short seconds_per_minute = 60,
            minutes_per_hour = 60,
            seconds_per_hour = seconds_per_minute * minutes_per_hour,
            hours_per_day = 24,
            CDT_offset = -5;

    const long seconds_per_day =
            static_cast<long>(seconds_per_hour) * hours_per_day;

    long seconds_today = time(nullptr) % seconds_per_day;

    short current_hour = seconds_today / seconds_per_hour,
            current_minute =
            seconds_today % seconds_per_hour / seconds_per_minute,
            current_second =
            seconds_today % seconds_per_hour % seconds_per_minute;

    current_hour += CDT_offset;

    short twelve_hr_clock_hour = current_hour % 12;

    string am_pm = "AM";

    if (current_hour >= 12) {
        am_pm = "PM";
    }

    cout << "\nThe current time is " << current_hour << ':';
    cout.fill('0');
    cout.width(2);
    cout << current_minute << ":";
    cout.width(2);
    cout << current_second << " (";
    cout << twelve_hr_clock_hour << ":";
    cout.width(2);
    cout << current_minute << ":";
    cout.width(2);
    cout << current_second << " " <<
         am_pm << ").\n\n";
}
```

mf98604@ares:~$ show-code complex.cpp

complex.cpp:

```
#include <iostream>
#include <cmath>

using namespace std;

class ComplexNum
{
    private:
        double real,
               imag;
        char symbol;

    public:
        void Input();
        void Output();
        void Assign(double real, double imag, char symbol = 'i');
        void Assign(ComplexNum x);
        ComplexNum Add(ComplexNum x);
        ComplexNum Subtract(ComplexNum x);
        ComplexNum Multiply(ComplexNum x);
        ComplexNum Divide(ComplexNum x);
        bool IsEqual(ComplexNum x);
        double Magnitude();
        ComplexNum Conjugate();
};
```

```
26
27  void ComplexNum::Input()
28  {
29      char plus_minus;
30      cin >> real
31          >> plus_minus
32          >> imag
33          >> symbol;
34      if (plus_minus == '-')
35      {
36          imag = -imag;
37      }
38      return;
39  }
40
41  void ComplexNum::Output()
42  {
43      cout << real;
44      if (fabs(imag) > 0)
45      {
46          cout << ((imag > -1) ? ('+') : ('-'))
47               << fabs(imag)
48               << symbol;
49      }
50      return;
51  }
52
53  void ComplexNum::Assign(double a, double b, char i)
54  {
55      real = a;
56      imag = b;
57      symbol = i;
58      return;
59  }
60
61  void ComplexNum::Assign(ComplexNum y)
62  {
63      real = y.real;
64      imag = y.imag;
65      symbol = y.symbol;
66      return;
67  }
68
69  bool ComplexNum::IsEqual(ComplexNum y)
70  {
71      return ((real == y.real) &&
72              (imag == y.imag) &&
73              (symbol == y.symbol));
74  }
75
76  ComplexNum ComplexNum::Add(ComplexNum y)
77  {
78      ComplexNum z;
79      z.real = real + y.real;
80      z.imag = imag + y.imag;
81      z.symbol = y.symbol;
82      return z;
83  }
84
85  ComplexNum ComplexNum::Subtract(ComplexNum y)
86  {
87      ComplexNum z;
88      z.real = real - y.real;
89      z.imag = imag - y.imag;
90      z.symbol = y.symbol;
91      return z;
92  }
93
94  ComplexNum ComplexNum::Multiply(ComplexNum y)
95  {
96      ComplexNum z;
97      z.real = (real * y.real) + (imag * y.imag);
98      z.imag = imag * y.imag;
99      z.symbol = y.symbol;
100     return z;
101 }
102
103 ComplexNum ComplexNum::Divide(ComplexNum y)
104 {
105     ComplexNum z;
106     z.real = (real * y.real + imag * y.imag) /
107              (y.real * y.real + y.imag * y.imag);
108     z.imag = -(real * y.real - imag * y.imag) /
109               (y.real * y.real + y.imag * y.imag);
110     z.symbol = y.symbol;
111     return z;
112 }
113
114 double ComplexNum::Magnitude()
115 {
116     return sqrt(real * real + imag * imag);
117 }
118
119 ComplexNum ComplexNum::Conjugate()
120 {
121     ComplexNum z;
122     z.real = real;
123     z.imag = -imag;
124     z.symbol = symbol;
125     return z;
126 }
127
128 int main()
129 {
130     cout << "\n\nPlease enter \'x\', a complex number of the form a + bi, \
131             "where a and b are real numbers and i is the square root of -1
132     ComplexNum complex_num_x;
133     complex_num_x.Input();
```

```
134
135       cout << "\n\nYou entered: \'";
136       complex_num_x.Output();
137       cout << "\'";
138
139       cout << "\n\nPlease enter \'y\', a complex number of the form a + bi, `
140               "where a and b are real numbers and i is the square root of -1
141       ComplexNum complex_num_y;
14
```