```
Script started on Wed 22 Feb 2012 12:06:47 AM CST
\033]0;georgia@georgia-MT6017: ~/cplusplus\007georgia@georgia-MT6017:~/cplusplus$ p
wd
/home/georgia/cplusplus
\033]0;georgia@georgia-MT6017: ~/cplusplus\007georgia@georgia-MT6017:~/cplusplus$ C
PP --version
This is CPP version 1.219 executing under perl v5.12.4 and compiling with:

g++ (Ubuntu/Linaro 4.6.1-9ubuntu3) 4.6.1
Copyright (C) 2011 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.


\033]0;georgia@georgia-MT6017: ~/cplusplus\007georgia@georgia-MT6017:~/cplusplus$ c
at proy033[K033[K033[K033[Kmidpoint.info
Name: Jakob Hansen
Class: CSC121 - Evening Section
Lab: Point to Point

Levels attempted:

1.5 - For just getting the basic instructions down.

+1 - For more natural manner of input



Program Description:

A program that will calculate the midpoint of a line defined by two coordinates
on a cartesian plane. The points must be entered in proper (x, y) notation.

\033]0;georgia@georgia-MT6017: ~/cplusplus\007georgia@georgia-MT6017:~/cplusplus$ c
at midpoint-done.cpp
#include <iostream>

using namespace std;

int main(void)
{
   double xa;
   double ya;
   double xb;
   double yb;
   char c;
   double midx;
   double midy;

   cout << "\n\t\tWelcome to the Midpoint Calculation Program!\n\n"
        << "To ensure accurate calculation, please enter the coordinates as: "
        << "(x, y)\n\n"
        << "Please Enter the first Point:";

   cin >> c >> xa >> c >> ya >> c;

   cout << "\nPlease Enter the second Point:";

   cin >> c >> xb >> c >> yb >> c;

   midx = (xa+xb)/2;
   midy = (ya+yb)/2;

   cout << "\nThe midpoint of the line segment bewteen "
        << "(" << xa << ", " << ya << ") and (" << xb << ", " << yb << ") "
        << "is (" << midx << ", " << midy << ")."
        << "\n\n\tThank you!\n\n";

   return 0;
}
\033]0;georgia@georgia-MT6017: ~/cplusplus\007georgia@georgia-MT6017:~/cplusplus$ C
PP midpoint-done.cpp
midpoint-done.cpp***


\033]0;georgia@georgia-MT6017: ~/cplusplus\007georgia@georgia-MT6017:~/cplusplus$ .
/midpoint-doen\033[K033[K033[Kne.out

                Welcome to the Midpoint Calculation Program!

To ensure accurate calculation, please enter the coordinates as: (x, y)

Please Enter the first Point:(5, 5)

Please Enter the second Point:(10, 10)

The midpoint of the line segment bewteen (5, 5) and (10, 10) is (7.5, 7.5).

        Thank you!

\033]0;georgia@georgia-MT6017: ~/cplusplus\007georgia@georgia-MT6017:~/cplusplus$ .
/midpoint\033[K-done.out

                Welcome to the Midpoint Calculation Program!

To ensure accurate calculation, please enter the coordinates as: (x, y)

Please Enter the first Point:(0, 0)

Please Enter the second Point:(8, 8)

The midpoint of the line segment bewteen (0, 0) and (8, 8) is (4, 4).

        Thank you!

\033]0;georgia@georgia-MT6017: ~/cplusplus\007georgia@georgia-MT6017:~/cplusplus$ .
/midpoint\033[K-done.out

                Welcome to the Midpoint Calculation Program!

To ensure accurate calculation, please enter the coordinates as: (x, y)

Please Enter the first Point:(-5, -5)

Please Enter the second Point:(5, 5)

The midpoint of the line segment bewteen (-5, -5) and (5, 5) is (0, 0).

        Thank you!

\033]0;georgia@georgia-MT6017: ~/cplusplus\007georgia@georgia-MT6017:~/cplusplus$ .
/midpoint-done.out

                Welcome to the Midpoint Calculation Program!

To ensure accurate calculation, please enter the coordinates as: (x, y)

Please Enter the first Point:(-5, 5)

Please Enter the second Point:(-5, -5)

The midpoint of the line segment bewteen (-5, 5) and (-5, -5) is (-5, 0).

        Thank you!

\033]0;georgia@georgia-MT6017: ~/cplusplus\007georgia@georgia-MT6017:~/cplusplus$ c
at midpoint.tpq
1. How many variables are needed here? (Hint: There are two points being
```

input and a third point being output. How many values are needed
to'make' a 2D point?)

Since you need two numbers for each point and two points are necessary
for there to be a midpoint, four variables are needed for the inputs.
Since there also needs to be a midpoint printed to the user, two more
variables are needed to hold the value of the result of the calculation.
If you want to get all fancy and make the user enter (,), then you need
one more char variable to hold that junk. So, uhh, 5 total in that case.

2. Would it make any sense to have whole-valued point coordinates (as opposed
to decimal-valued point coordinates...)? That is, would it ever be correct
for you to assume that a point's coordinates would never have fractional parts?

It would depend on the application... Since there was no explicit limitation
in the assignment (at least not that was noticed by the present author) it
makes sense in this abstract environment to allow for the
possibility of cooridinates being real rather than whole numbers. But, if
the program was used in some endeavor dealing with some kind of units that
were indivisible (or practically indivisible) like, I dunno, legos or
paving tiles it might make sense to limit the input in that way. It seems
like fractional values will be such a common output, though, that anyone using
this equation would probably need to input fractional parts from time to time.

3. What is the formula to calculate a midpoint? Compare/Contrast the x part of
the formula and the y part. (I.E. How are they similar? How are they different?)

For two points (x1,y1) and (x2,y2)

$$\text{The midpoint is: } \frac{x1+x2}{2}, \frac{y1+y2}{2}$$

4. What happens if the input points are in different quadrants? In the same
quadrant? On an axis? At the origin? (I.E. Can your program ever go wrong
because of the points' locations?)

The formula will give the correct result regardless of the sign of any of
the coordinates. Testing different combinations bears this out.

5. How many different ways can you pick a pair of points from the Cartesian
plane: quadrant(s), axis/axes, origin? (In other words, how many tests would
it take to completely test your program?) (You do not have to script that many
tests. But you should be able to calculate how many there should be. That way
when you are working in industry, you can more thoroughly test your
applications and run Micro-sloth out of business. *chuckle* *grin*)

I believe this is a COMBINATION problem...so, there are 9 "options":
origin, x axis, -x axis, y axis, -y axis, somewhere (x,y), somewhere (-x,y),
somewhere (x,-y), somewhere (-x,-y).

So the potential combinations of two points on any two potential "zones"
is 36.

1. What datatype for characters in notation?

Char. Not sure how to be less terse than that!!

2. How many variable for fancy input?

At most 3, which I did at first, but then realized you only need one.

3. How does cin know when a character stops and a number starts
well because of the declaration of the variables - cin expects a 'single char'
and when it finishes with that it goes on to read in the numbers (if thats
what it expects) until it hits another char or '\n'.

4. what happens with non-parentheses or comma
The single character that is typed where cin expects a char will be stored as
a char, so if you type a number that has two digits, the first one will be
stored into the char variable and only the second digit will be read as a number
this could be bad.

5. What if everything is non-numeric?

Again, wherever cin expects a char, it stores whatever is typed as a char. If
then you go along and type a letter or a symbol where cin expects a number,
you'll get all kinds of ascii garbage numbers.

\033]0;georgia@georgia-MT6017: ~/cplusplus\007georgia@georgia-MT6017:~/cplusplus$ e
xit
exit

Script done on Wed 22 Feb 2012 12:08:47 AM CST