Dalhousie University
CSCI 2132 — Software Development
Winter 2018
Assignment 6

Distributed Wednesday, March 7 2017.
Due 9:00PM, Wednesday, March 21 2018.

Instructions:

1. The difficulty rating of this assignment is *gold*. Please read the course web page for more information about assignment difficulty rating, late policy (no late assignments are accepted) and grace periods before you start.

2. Each question in this assignment requires you to create one or more regular files on bluenose. Use the exact names (case-sensitive) as specified by each question.

3. C programs will be marked for correctness, design/efficiency, and style/documentation. Please refer to the following web page for guidelines on style and documentation for C programs of median size (which applies to this assignment):

   `http://web.cs.dal.ca/~mhe/csci2132/assignments.htm`

   You will lose a lot of marks if you do not follow the guidelines on style and documentation.

4. Create a directory named `a6` that contains the following file (this is the file that assignment questions ask you to create): `a6q1.c`. Submit this directory electronically using the command `submit`. The instructions of using `submit` can be found at:

   `http://web.cs.dal.ca/~mhe/csci2132/assignments.htm`

5. Do NOT submit hard copies of your work.

As mentioned in the first lecture of this course, this assignment is a gold-level assignment. It also has a small number of bonus marks, to reward people who work hard on assignments of gold level: Recall that we have the best 6 out of 7 policy for assignments.

Since this assignment is supposed to be more difficult than previous assignments, start working on this assignment early. As usual, the instructor and the TAs are there to help.

To help you avoid losing marks, a draft of the marking scheme for the first question can be found at (this is subject to minor changes); review carefully as this is the first assignment that requires you to organize your program using functions:

`http://web.cs.dal.ca/~mhe/csci2132/assignments/a6q1.pdf`

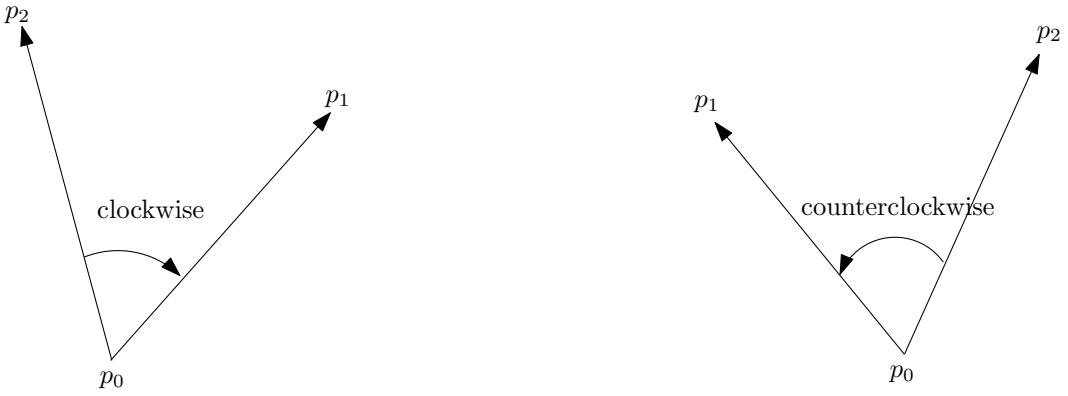

Figure 1: Clockwise vs Counterclockwise.

Background: This problem asks you to compute the convex hull of a point set. This is a fundamental problem in computational geometry, and it has many applications in GIS and graphics applications. It is even used in software that many of you have tried before. Think of a vector graphics editor or even the feature of drawing objects in Powerpoint. These programs often let you group shapes (polygons, line segments, etc.) that you draw as one entity. Then, you can click on any shape in this group to move the entire group around. How does the software detect whether you clicks over any shape in this group? One common solution is to compute the convex hull of this group, and then the software simply tests whether you clicks inside the convex hull (which is a convex polygon as we will see soon), and this can be done very efficiently.

As some background knowledge in geometry and how to perform computations in geometric data sets efficiently is required, the first part of the assignment is focused on such background knowledge, before the problem specifications of the programming questions.

Properties of directed line segments: Recall (from what you have learned in geometry lessons before) that a line segment is a part of a line that is bounded by two distinct end points and contains every point on the line between these two end points. A *directed line segment* is a line segment endowed with the additional property of direction.

Let $p_1 = (x_1, y_1)$ and $p_2 = (x_2, y_2)$ be two distinct points on the plane. That is, (x_1, y_1) is the coordinates of p_1 and (x_2, y_2) is the coordinates of p_2. We then use $\overrightarrow{p_1p_2}$ to represent the directed line segment from p_1 to p_2.

Given two directed line segments that share the starting endpoint, a useful operation is to determine which line segment is clockwise from the other. More precisely, we are given three points $p_0 = (x_0, y_0)$, $p_1 = (x_1, y_1)$ and $p_2 = (x_2, y_2)$, and these three points do not lie on a single straight line. We would like to find out whether $\overrightarrow{p_0p_1}$ is clockwise or counterclockwise from $\overrightarrow{p_0p_2}$. That is, if we would like to turn $\overrightarrow{p_0p_2}$ by less than 180 degrees toward $\overrightarrow{p_0p_1}$, so that they will align, should we turn clockwise, or counterclockwise? Figure 1 gives two examples. In the first example, $\overrightarrow{p_0p_1}$ is clockwise from $\overrightarrow{p_0p_2}$. In the second example, $\overrightarrow{p_0p_1}$ is counterclockwise from $\overrightarrow{p_0p_2}$.

To find this out, we evaluate the following expression:

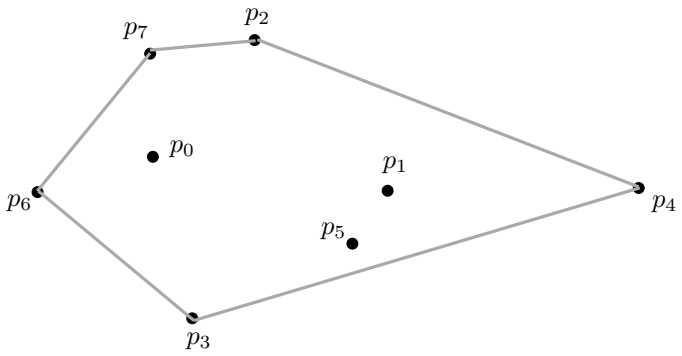

Figure 2: An example of convex hull.

$$(x_1 - x_0)(y_2 - y_0) - (x_2 - x_0)(y_1 - y_0)$$

If the result is positive, then $\overrightarrow{p_0p_1}$ is clockwise from $\overrightarrow{p_0p_2}$. If negative, it is counterclockwise.

This approach is based on cross products. Here I simply gave you the formula, and if you are interested, you could do some research to learn more about cross products. This approach is cool because it does not require division at all. This is desired as divisions are typically slower than addition, subtraction or multiplication on most CPUs, and to perform division, we also need avoid division by zero.

Convex Hull: For a given point set P, its convex hull is the smallest convex polygon C for which each point in P is either inside C or on the boundary of C. Figure 2 gives an example. In this example, $P = \{p_0, p_1, p_2, p_3, p_4, p_5, p_6, p_7\}$. The convex hull of P is the convex polygon defined by p_2, p_4, p_3, p_6 and p_7.

How to compute the convex hull efficiently? There are many algorithms and here you are asked to implement an algorithm called *Jarvis's march*. To introduce this algorithm, we will assume that the x-coordinates of points are distinct, and the y-coordinates of points are also distinct. We will also assume that no three points in the point set lie on a single line.

Take Figure 2 as an example. It is clear that the point with the largest y-coordinate (p_2 in this example) is one of the vertices of the convex hull. Then, use a pencil to draw directed line segments $\overrightarrow{p_2p_0}, \overrightarrow{p_2p_1}, \overrightarrow{p_2p_3}, \cdots, \overrightarrow{p_2p_7}$ (Be sure to do this yourself). We know that p_4 is the vertex next to p_2 on the convex hull in clockwise order. What is special about $\overrightarrow{p_2p_4}$ among all the line segments that we drew? I will give the answer in the next paragraph; think about this yourself for a while to see if you can find this out by yourself.

What is special about $\overrightarrow{p_2p_4}$ is that it is counterclockwise from any other directed line segment that we drew.

Now, consider the point p_4 which is the next vertex on the convex hull after p_2 in clockwise order. We first erase the line segments that we drew before, and then draw all the direct lines segments $\overrightarrow{p_4p_0}, \overrightarrow{p_4p_1}, \overrightarrow{p_4p_3}, \overrightarrow{p_4p_5}, \overrightarrow{p_4p_6}, \overrightarrow{p_2p_7}$. We can observe the same property for $\overrightarrow{p_4p_3}$, which leads to the next vertex p_3.

We can repeat this process until we are back at vertex p_2. By now we have computed the convex hull. Jarvis's march is based on this idea. The following is the pseudocode that

computes the convex hull, C, of the point set $S = \{p_0, \cdots, p_{n-1}\}$:

1: Let p_t be the topmost point in S
2: $C[0] \leftarrow t$
3: $i \leftarrow 0$
4: **repeat**
5: Find out the point p_k such that $\overrightarrow{p_{C[i]}p_k}$ is counterclockwise from any $\overrightarrow{p_{C[i]}p_j}$ where $j \neq C[i]$ and $j \neq k$.
6: $i \leftarrow i + 1$
7: $C[i] \leftarrow k$
8: **until** $k == t$
9: Do not include $C[i]$ when using C to report the convex hull of S, so that the topmost point will not be reported twice.

Questions:

1. [30 marks] This question asks you to implement Jarvis's march to compute the convex hull of a given point set.

   Note that if your solution does not implement Jarvis' march at all, then up to only a very small number of marks can possibly be awarded for handling error cases, no matter whether your program can compute convex hull correctly or not.

   General Requirements: Use `a6q1.c` as the name of your C source code file.

   We will use the following command on bluenose to compile your program:

   ```
   gcc -std=c99 -o hull a6q1.c
   ```

   As many numbers have to be read from stdin to test your program, you are expected to use input redirection to read the input from a file. More precisely, you can run your program using the following command:

   ```
   ./hull < a6q1.in.0
   ```

   In the above command, the file `a6q1.in.0` is a plain text file. The first line of this file is an integer that is the number of points in the point set. Starting from the second line, each line contains two integers. The first integer is the x-coordinate of a point, and the second integer is its y-coordinate. Hence, in this question, you can assume that coordinates are `int` values.

   To simplify your work, you can assume that the x-coordinates of points are distinct, and the y-coordinates of points are also distinct. You can also assume that no three points in the point set lie on a single line.

   For example, the following file `a6q1.in.0` contains a set of points:

```
12
5 19
33 2
-5 88
54 5
12 13
18 39
15 42
17 -5
-3 23
9 29
-8 17
-1 25
```

Your program should then find the convex hull of these points. It first prints the number of vertices of the convex hull. It then report the coordinates of these vertices in clockwise order starting from the topmost vertex. To report a vertex, print its x-coordinate and y-coordinate in a single line, use exactly one space between these two values (this is the only space character in this line), and terminate this line with a newline character. For example, the output of the above example is expected to be

```
4
-5 88
54 5
17 -5
-8 17
```

Error Handling: Your program should print an appropriate error message and terminate in each of the following cases:

(a) The number of points is less than 3 (obviously you need at least 3 points to define a polygon).

(b) When you try to read an integer from the input file, you encounter an illegal character. For example, your `scanf` function encounters `a100` in the input file and cannot read it as a decimal number.

If there is more than one problem with user input, your program just has to detect one of them. You can assume that everything else regarding the input file is correct.

Testing: To help you make sure that the output format of your program is exactly what this question asks for, several files are provided in the following folder on bluenose to show how exactly your program will be automatically tested:

`/users/faculty/prof2132/public/a6test/`

In this folder, open the file `a6q1test` to see the commands used to test the program with three test cases. The output files, generated using output redirection, are also given.

To check whether the output of your program on any test case is correct, redirect your output into a file, and use the UNIX utility `diff` (learned in Lab 3) to compare your output file with the output file in the above folder, to see whether they are identical.

Since these output files are given, we will **apply a penalty to any program that does not strictly meet the requirement on output format**. This includes the case in which your output has extra spaces, or has a missing newline at the end. Note that **it is extremely important to use diff here, as the number of values in the output file is large**.

We will use different test cases to test your program thoroughly. Therefore, construct a number of test cases to test your program thoroughly before submitting it.

Hints:

(a) If you use `gdb` to debug your program, the following command can run your program with input redirection (say, the input file is named `a6q1.in.0`):

```
run < a6q1.in.0
```

(b) If you use variable length arrays to store point coordinates, then, to print the content of the array, you have to use the command given in 4(d) of Lab 6.

(c) To make it easier to test your program, you could make use of some free online programs for graph plotting. There are many online, and I found the following one: `http://itools.subhashbose.com/grapher/index.php`

These programs will not draw convex hulls, but they will make it easier for you to see which points are on the convex hull. Alternatively, you could buy engineering pads and draw points on them.

2. [5 marks extra credit only] **V**ERY IMPORTANT: Before you work on this question, make a backup copy of `a6q1.c`. This is because this question asks you to modify this program. Make a backup copy so that you will not lose your work for Question 1 should you somehow introduce a bug that you are unable to fix before the deadline.

   In this bonus question, you are asked to improve the implementation of your program by modifying it. Instead of creating a new source file, modify the program you wrote for Question 1.

   Since during this process, you might introduce more bugs which you may or may not have sufficient time to eliminate, you are strongly recommended to fully test your program for Question 1 first and use the `submit` command to submit one version. If you use `git` to manage your source code (see Lab 7, though you are not required to use it), also commit one version. If you do not use `git`, at least make a copy of your

program and store it in a different folder before modifying it. When you finish, simply submit `a6q1.c` again.

In this question, You are asked to modify your program so that it will still compute the convex hull correctly even if x and y-coordinates of points are not necessarily distinct (2.5 marks), and there might be groups of three or more points in the point set that lie on a single line (the other 2.5 marks). You can assume that there are no duplicate points in the given point set.

Recall that you are required to report vertices of the convex hull. Thus, if a point from the point set is on the boundary of the convex hull, but it is not a vertex, then do not report it.

Note that for this assignment, the header file `math.h` is not required.

Be sure to fully test your program before you submit. Also use the sample input/output given for Question 1 to make sure that your output format is correct.

Make sure to provide sufficient documentation, as documentation marks for Question 1 will be awarded according to the entire program you submit (see the marking scheme). That is, the 5 bonus marks for this assignment are for correctness only.