**CS 133U: C Programming**

# Assignment 4: Jelly Bean Statistics

| **Academic Integrity** **You may NOT, under any circumstances, begin a programming assignment by looking for completed code on StackOverflow or Chegg or any such website, which you can claim as your own. Please check out the** [**Student Code of Conduct at PCC.**](https://www.pcc.edu/student-conduct/conduct/quick-view-of-policy/)  The only way to learn to code is to do it yourself. The assignments will be built from examples during the lectures, so ask for clarification during class if something seems confusing. If you start with code from another source and just change the variable names or other content to make it look original, you will receive a zero on the assignment.  I may ask you to explain your assignment verbally. If you cannot satisfactorily explain what your code does, and answer questions about why you wrote it in a particular way, then you should also expect a zero. |
| --- |

## 

Sometimes you will see an activity at a fair where people are asked to guess how many jellybeans there are in a jar. Sometimes prizes are given to the person who guesses a number closest to the actual number. You can use a little math and write a program to solve this problem for any size jelly bean and jar!



## Purpose

The purpose of this assignment is to develop an algorithm and a working C program, to calculate the number of jellybeans in a jar, given the dimensions of a jellybean, and the capacity of the jar.

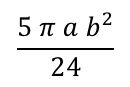
Then you will output some statistics based on the information the user has entered. You will be entering in jar and jellybean dimensions, and you will continue asking the user for jar and jellybean dimensions until they are done. While the user wants to add more input, you will add the data to **accumulators** (an accumulator is a variable the program uses to calculate a sum or product of a series of values) to perform statistical analysis after they are done.

After completing this assignment you will be able to:

* Use a do while loop to prompt the user for at least one set of jar and jellybean dimensions
* Use an accumulator to keep track of the total number of jellybeans all the jars can hold, and then number of entries entered by the user
* Use a min/max algorithm to determine the largest jellybean and jar size that contains the largest estimate of jellybeans
* Calculate the average number of jellybeans the jars can hold

## Task

* Based upon hours of research, zillions of jelly beans, and a bunch of mason jars, it has been concluded that the volume of a jelly bean can be roughly modeled as the average of the volume of the exterior bounding cylinder and of the volume of the interior bounding ellipsoid.
* Our formula for the volume j of a single jelly bean is:



where a and b are respectively the length and height of the jelly bean.

* Exhaustive research has determined the loading factor — the percentage of a jar that can be occupied by jelly beans — is constant, with the constant being 69.8% (i.e., .698).
* Your task is to produce a program that separately prompts and reads three values, their order being the average length and height of a jelly bean (decimal values) and the size of a jar (integer value). The dimensions are in centimeters and the volume of the jar is in mLs (note: one mL equals one cubic centimeter).
* Open the [Algorithmic Design Document](https://docs.google.com/document/d/1Oyww4GMtCmbZL3Xq_JNw8wKtnUzGsG2ebZeEwxqwOcs/edit?usp=sharing), make a copy, and follow the steps to create your algorithm.
* You must express your algorithm as **pseudocode** or a **flowchart.**
* Print a welcome message for your program.
* When the program runs, you must read the jelly bean length (cm) and the jelly bean height (cm), separated by space exactly as shown in the sample run below. **This is part of the required coding construct.**
* Then read the jar volume (mL, int data type) on the next line.
* Find the volume of the jelly bean using the formula above.
* The jellybeans will take up 69.8% of the jar’s total volume. Create a constant called LOAD\_FACTOR and set it to 0.698.
* The estimate for the number of beans that can fit in the jar is

jarVolume \* LOAD\_FACTOR / jellyBeanVol

* First do this calculation on paper and pencil and make sure you get the same answers as my two sample runs shown below in the Criteria for Success.
* **To make your estimate as accurate as possible, use the built-in constant value defined in the “math.h” library. You MUST use this code to define a constant named PI exactly as defined below.**

**#include <stdio.h>**

**#include <math.h>**

**#define M\_PI 3.14159265358979323846**

**const double PI = M\_PI;**

**int main() {**

**printf("%lf\n", PI);**

**}**

* Output the number of jelly beans (integer) that will fit in the jar, do not try to round your decimal estimate. You just need to use an integer variable for the bean count. The extra jelly bean produced by rounding would not fit in the jar. See sample calculation in the sample runs below.
* Prompt the user if they want to run the program again.
* When the user is finished entering jelly bean and jar information:
  + Display the number of entries entered.
  + Calculate and display the average number of jelly beans for the number of jars entered.
  + Display the largest jelly bean size entered.
  + Display the jar size that contains the largest number of jelly beans (careful - this is not the largest jar, but the size of the jar that had the largest estimate!).
* Print a goodbye message.
* **Do not use any return or break statements inside your while loop.**
* **Use only the concepts we have learned so far.**

## Criteria for Success

* Look at the sample run below. Determine the variables you will need for the various calculations (your list may include other variables):
  + Single jelly bean size (length and height)
  + Single jar size
  + Estimate number of jelly beans for the given jar size
  + Total number of entries by user
  + Total number of jelly beans for all jars entered (accumulator)
  + Largest jelly bean entered (use max algorithm, see additional resources section below)
  + Largest number of jelly beans in a jar (use max algorithm)
  + Jar size that contains the largest number of jelly beans (use max algorithm)
* **ALERT!!** Look at the sample run below. Notice you are finding the jar size that holds the maximum number of jelly beans, **NOT** the maximum jar size!!
* Test your program using the following sample runs, making sure you get the same output when using the given inputs (in **blue**):

| Welcome to my Jelly Bean Guesser program!  Enter the length of a jelly bean (cm) and the height of a jelly bean (cm) separated by a space: **1.52 0.9**  Enter the jar size (ml): **500**  Estimate of jelly beans in the jar: 433  Do you want to enter more (y/n): **y**  Enter the length of a jelly bean (cm) and the height of a jelly bean (cm) separated by a space: **2.0 1.0**  Enter the jar size (ml): **25**  Estimate of jelly beans in the jar: 13  Do you want to enter more (y/n): **y**  Enter the length of a jelly bean (cm) and the height of a jelly bean (cm) separated by a space: **1.9 1.3**  Enter the jar size (ml): **250**  Estimate of jelly beans in the jar: 83  Do you want to enter more (y/n): **n**  Number of entries: 3  Average number of jelly beans: 176.33  Largest jelly bean: 2.10 cm^3  Jar size for largest jelly bean estimate: 500 mL  Thank you for using my program! |
| --- |

* Complete zyBooks section **CS133U 4. Loops** activities.
* Ensure required coding conventions for this assignment are met:
  + Jelly Bean height and length must be read in on a single line
  + M\_PI must be defined as specified in the description above
  + No gotos or other jump-style programming are allowed.
  + Breaks statements are only allowed as part of a switch.
  + Only concepts from Weeks #1-4 can be used.
* Complete all sections of your Algorithmic Design Document.
  + Include **pseudocode** or a **flowchart** in part d of the design document.
  + Include at least one screenshot of your code successfully running under Sample Runs.
* Please open and compare your work with the [grading rubric](https://docs.google.com/document/d/14vwZLlN8HzHJiyj_q7xX_PFHR_do2AT5LO5amsc5DD0/edit?usp=sharing) before submitting.
* Remember to follow all [style guidelines](https://docs.google.com/document/d/1jqCJOM_nxvWalw9EbG9Rk44wj8SnhlektNulzzYfNw0/edit?usp=sharing).
* Download your Algorithmic Design Document as a PDF (File -> Download -> PDF), rename it to a04.pdf.
* Name your C source file a04.c and upload with a04.pdf to the D2L assignment.
* Do your own work. Consult the syllabus for more information about academic integrity.

## Additional Support

* [Finding the maximum and minimum from user input](https://youtu.be/RwqEqqqU9QI) video.
* Post a question for the instructor in the Ask Questions! area of the Course Lobby.