# Academic honesty and plagiarism

- Plagiarism is unacceptable.
  - Code you submit must be your own. No copying, adapting, or submitting code you did not create is allowed. Working together and presenting variants of the same file is not acceptable. Here are some specific guidelines to make sure you don’t cross the line:
    - Do not exchange programs or program fragments in any form on paper, via e-mail, photos, or by other means.
    - Do not copy solutions from any source, including the web or previous quarters’ students.
    - Do not discuss code with other students at the level of detail that will lead to identical programs or program fragments.
  - Ask me if you are uncertain.
  - **All violations of the academic honesty will be immediately referred to the dean's office.**
- **Plagiarism detection will be applied to all code. It is very good and will catch you!! (20% of a class referred to dean!!)**

Lab 7: Computer Vision, Due May 26, 7pm

The task for this assignment is to implement a set of shape detection algorithms. Your functions will receive a 2D structure of size 200x200 representing the intensity of pixels of 200x200 pixel image and it should return information about the content of if.

Please use python3. Here you are only allowed to import “common” and “math” in your student code, no other imports are allowed

Problem 1: slope-intercept equation line detection
You have to develop a program to detect lines in images. Luck is in your side and in this exercise all lines will be black with a white background, so edge detection is not needed. The biggest problem you will find is some random figures and noise in the image.

For this task you are to complete one function: def detect_slope_intercept(image):

The function receives:

- A 2d structure called image[y][x] containing the pixel intensities of the image.

The function must return:

- a Line structure (see common.py) with the m and b fields containing the parameters of the line function (mx+b=y).

Considerations:

- The voting space should be size exactly 2000x2000 “bins”
- m is bounded to -10 and +10
- b is bounded to -1000 and +1000
- The margin for correctness will be +/-.1 for m and +/-3 for b during grading

Problem 2: counting quarters
For this problem you have to create a program to count the number of circles in an image. All circles have the same radius (in pixels) and they will be complete (no overlapping between circles or partially outside the image)

For this task you are to complete one function: def detect_circles(image):

The function receives:

- A 2d structure called image[y][x] containing the pixel intensities of the image.

The function must return:

- an integer with the number of circles detected in the image.

Considerations:

- The voting space should be size 200x200 (size of the image)
- The radius is always 30 (all circles have the exact same footprint)