Quiz 2

Instructions:

- You are allowed to access the internet as well as any files you may have on your computer. Feel free to copy solutions from the internet. This is at your own risk – if you copy a solution and it does not work, we will not give you points just because some website said it was correct.
- You will need to submit a single PDF file on gradescope, as well as some python files (in the usual way – as a zipped folder named Firstname_Lastname_Exam1.zip on the classroom). You should submit on gradescope first. That's the time we will use to check if you submitted late.
- You must ALSO paste the code you write into the file you submit on gradescope. So you'll submit the code twice – once pasted within your PDF, and again as a python file via classroom.
- The first page must **only** contain:
  - Your name
  - Your Ashoka email
  - Today's date
  - The words "FCP Quiz 2"
- Each solution (not sub-part) must start on a new page. Paste the code for coding questions.
- You have 90 minutes. We will release 10 minutes early, and the submission time is 10 minutes late to allow for uploading/downloading issues. Do not spend too much time on any single problem. Read them all first, and attack them in the order that allows you to make the most progress.
- You are NOT required to answer all the questions. I fully expect MOST of you to skip MULTIPLE questions! If you are unable to upload everything on time, email your solutions to your TA.

Question	Points
I Know What You'll Do Next Summer	25
What's in a thought?	25
The Second Order	20
Seeing is Believing	65
Hearing Backwards	25
Do not sort!	20
Total:	180

Problem 1. [25 points] **I Know What You'll Do Next Summer** (1 part)
Read the wikipedia page on Newcomb's Paradox. Make sure you read the "consciousness" section.

Analyze the paradox and state what you would do as a player and why.

Problem 2. [25 points] **What's in a thought?** (2 parts)

(a) [10 points] Look closely at these faces.

Figure 1: Image generated by a generative adversarial network - StyleGAN2 (Dec 2019)

These are three realistic images of faces of people: none of these people exist in real life; these pictures are generated by computers. It seems that machines have started to create things that seem real! Imagine what machines will be able to do in 10 years? 50 years? Machines may soon be able to generate world-class music, paintings, and novels. This is a slightly different thing than, say, an AI that can diagnose diseases better, etc. What do you think are the implications of such machines that seem creative?

Links to check out:
https://www.nytimes.com/interactive/2020/11/21/science/artificial-intelligence-fake-people-faces.html

https://www.theguardian.com/commentisfree/2020/sep/08/robot-wrote-this-article-gpt-3

(b) [15 points] On page 5, there is a cartoon with two robots having the following conversation (I am transcribing the conversation here):

Robot 1: Do you think humans really have feelings, or are they just programmed to act like they do?

Robot 2: Programming, obviously. Robots are created all at one, designed to experience pure emotional states, just for the sake of having them. Humans were designed by evolution to use emotions to make heuristic decisions and manipulate other social beings.

Robot 1: But they really think they're feeling emotions.

Robot 2: Uh huh. You know how you feel bad whenever someone dies?

Robot 1: Of course. Every second or so.

Robot 2: Humans only feel bad when humans they know die. Especially ones who are kin.

Robot 1: Genetic programming coupled to CPU resource maximization.

Robot 2: Bingo.

Link to original comic:https://www.smbc-comics.com/comic/do-humans-have-feelings

Answer the robot's question – do humans really have feelings, or are we just programmed to act like we do? If a robot was a sufficiently good actor so as to fool all humans into believing it had feelings – to even "fool itself", would that be the same as having feelings?

Problem 3. [20 points] **The Second Order** (3 parts)
CODE! We have decided to perform a new kind of sorting: we call it, **AlmostSort**.

(a) [1 point] First, generate a list with n integers $(0, 1, 2, \ldots, n-1)$, shuffled into random order.

(b) [9 points] Sort the elements at even and odd *index positions* separately. An example of such a sorted list might be: 6, 0, 7, 1, 8, 2, 9, 3, 10, 4, 11, 5. Notice how all the odd numbered positions are sorted relative to each other (same with the even index positions).

(c) [10 points] Now, take a number k (this may be user input or just a variable that we can set to any number, e.g., 3). Every chunk of k items in the array from the previous part should now be sorted. So, for example: 0, 6, 7, 1, 2, 8, 3, 9, 10, 4, 5, 11 would be the output using $k = 3$.

Problem 4. [65 points] **Seeing is Believing** (7 parts)
Start by downloading a set of 2-3 images from the internet for testing. We will be using our own images for testing. Your images should NOT be square. *Note: If you are unable to use an image library, you can use two-dimensional integer arrays. Beware! Numpy arrays and normal integer arrays behave a little differently when slicing, so make sure you state which one you are using and stick to that style.*

(a) [5 points] You don't have to write code for this: describe how, given an angle X, you would rotate the image counterclockwise by that angle.
Limit: a couple lines. If it is obvious you copied without understanding, we won't give you points for this.

(b) [5 points] Describe a kernel for edge detection and explain why it works. Why do the numbers have to add up to 0? Can we write an edge detector where this is not the case?

(c) [7 points] CODE! Write a program to invert the colours in an image (you just need to do 255-x for each colour; google this). Then, run one of the edge detection kernels.

(d) [7 points] After you write the program above, do you think the results should change with / without inverted colours? Why or why not? Explain.

(e) [16 points] Imagine a 2x2 kernel! (Bottom right corner of the kernel is the "main" pixel.) Think about what happens with the following kernels and explain why! (4 points each)

```
 1  0
 0 -1

-1  0
 0  1

-1 -1
 1  1

 1 -1
 1 -1
```

(f) [15 points] Add a top-left to bottom-right, blue "gradient filter" to the image. That is, your image should have a blue overlay, starting from totally blue to the top-left of the image, slowly fading away diagonally as we go towards the bottom right. Output as "blue.jpeg".

(g) [10 points] Add a black border which is k pixels wide.

Problem 5. [25 points] **Hearing Backwards** (2 parts)
You may use your own music or any of the songs we shared with you for this; you may decide your own sampling rate. *Note: If you are unable to use a sound library, you can use integer arrays. Beware! Numpy arrays and normal integer arrays behave a little differently when slicing, so make sure you state which one you are using and stick to that style.*

(a) [15 points] Reverse every k seconds of the music (e.g., $k = 1.5$ should mean every 1.5 second chunk of the song is reversed). Your output file should be named "k-rev.mp3".

Be careful about the ending! If the song ends and you are left with a smaller-than-1.5-second chunk, you should reverse that also.

(b) [10 points] Now, add a $k - second$ reverse echo to this song: that is, you should hear an echo of the sound from k seconds *after* the current time. (Just like a normal echo is a sound from some time before the current time.)

EXTRA CREDIT (5 points): If you had done these two parts in reverse order, would the result sound the same? Justify your answer stating which index positions would get added to which other index positions.

Problem 6. [20 points] **Do not sort!** (2 parts)
Allow the user to input an array. You need to pair up biggest and smallest elements (each pair should have the bigger number first) and output. For example, if the input A=[8,1,2,3,4,5,6,7], then, you should output 8,1,7,2,6,3,5,4. You are allowed to use python's inbuilt sorting function or any other.

Internally, what you're doing is this:
(biggest and smallest, with the bigger one first) 1 and 8 $\implies$ 8,1
(next biggest and smallest, with the bigger one first) 2 and 7 $\implies$ 8,1,7,2
(and so on) 3 and 6 $\implies$ 8,1,7,2,6,3
4 and 5 $\implies$ 8,1,7,2,6,3,5,4

(a) [10 points] PSEUDOCODE! Write down how to solve this in plain English "code".

(b) [10 points] CODE! Write a program that implements this.

EXTRA CREDIT Problem A. [5+10=15 Points] **Squaring things away** (1 part)
PSEUDOCODE! For 5 points, write a pseudo code for a program that which when given an image of size A $\times$ B can resize it to C $\times$ D, where the latter two are inputted by the user. Explain clearly why this works in a couple of lines. Then for 10 more points, CODE! Code it up. Make sure your code works for an edge case like C=1.

EXTRA CREDIT Problem B. [5 Points] **Silhouette** (1 part)
We used a modified sharpening kernel to detect edges in class. How would you modify a blurring kernel to detect edges?


DO YOU THINK HUMANS *REALLY* HAVE FEELINGS, OR ARE THEY JUST PROGRAMMED TO ACT LIKE THEY DO?
PROGRAMMING, OBVIOUSLY.
ROBOTS ARE CREATED ALL AT ONCE, DESIGNED TO EXPERIENCE PURE EMOTIONAL STATES, JUST FOR THE SAKE OF HAVING THEM.
HUMANS WERE DESIGNED BY EVOLUTION TO *USE* EMOTIONS TO MAKE HEURISTIC DECISIONS AND MANIPULATE OTHER SOCIAL BEINGS.
BUT THEY REALLY *THINK* THEY'RE FEELING EMOTIONS.
UH HUH.
YOU KNOW HOW YOU FEEL BAD WHENEVER SOMEONE DIES?
OF COURSE. EVERY SECOND OR SO.
HUMANS ONLY FEEL BAD WHEN HUMANS THEY KNOW DIE. ESPECIALLY ONES WHO ARE KIN.
GENETIC PROGRAMMING COUPLED TO CPU RESOURCE MAXIMIZATION.
BINGO.
smbc-comics.com