**TASK1 :**

I started by including the stdint.h header file right after stdio.h. The reason for using this is to use some different sort of standard integer data types like we have used right after (uint8\_t).Here, "u" stands for unsigned , uint8\_t type is an unsigned 8-bit integer, "int" for integer, and "8" for 8 bits. For simplicity, I created an alias called BYTE for this type.Next, I set declared two file pointers: one for the source image file (100dollars.tif) to read from, and another for a new destination file where the flipped image will be saved. Once everything is done correctly, we'll find the flipped image saved as backup\_100dollars.tif in the same folder. Here, I have used rb and wb respectively for reading and writing in binary mode.To keep the image header intact, I created an array called metadata to store the first 8 bytes from the source file , because we need to keep the heading bytes intact. These bytes were then written directly to the destination file. After that, I checked the height and width of the source image and created a 2D array to hold the pixel data.To flip the image, I looped through each row in reverse order, which effectively mirrored the image horizontally. I then used fwrite() to write this flipped pixel data to the destination file.To handle the remaining metadata, I created another array called remMetadata. This array read the rest of the source file until the end and then wrote that data to the destination file using fwrite(). And that’s how the flipping process was completed.Finally, to clean things up and free system resources, I closed both the source and destination files using fclose().

**TASK2 :**

Firstly, I added the stdint.h header file after the stdio.h header file. The reason for using this is to use some different sort of standard integer data types like we have used right after (uint8\_t). Here u stands for unsigned, int stands for integer data type and 8 means that this data type consists of 8 bits. We have created an alias BYTE for this. Then we declared two file pointers where one directs to the source image (100dollars.tif) for reading purpose and the other one directs to a non-existing file, where the blurry image will be stored and after the completion of both these activities properly, a flipped image of the file name blurry\_100dollars.tif will be created in the folder. Here,I have used rb and wb respectively for reading and writing in binary mode. Next up, I have created metadata array which stores the first 8 bytes from the source file, because we need to keep the heading bytes intact. Then, we wrote those 8 bytes in the destination file. Then we saw the height and width of the sample image and then created an 2D array to store the px data. For applying the blurry effect, the idea was to traverse through each row and get the pixel values of the adjacent 4 horizontal pixels and then calculate the average value and then applying the average value to each group of the 4 pixels. A new variable named avg was created and it simply meant sum/4, beforehand we declared sum=0. The main job is done here as the pixel values are changed. Now, for copying the remaining metadata, another array named remMetadata was created which read from the source file until it finished and then we stored that into the destination file using fwrite().  Finally, to release system resources and to finalize file operations we closed the source and destination file using fclose().