**Low Pass Filter**

**Name1:** Ahmed Saad Ibrahim

* **Idea :-** 
  + Master rank read the data.
  + Split it on all ranks include master rank.
  + We untie the related between the data.
  + Send to each rank subarray contain its data that will process on it.
  + Send to each rank some related data like recvcount, image height, process length.
  + We work on 1d array data and when we sub it work also on 1d array.
  + Every rank will process the data and send it to the master rank.
  + Master rank collect the data in the main image data.
* **How to split the data :-**
  + Split the image data by height on all ranks.
  + We added to the last rank the reminder height.
  + Every rank take its height\*image width this will get the process length.
  + We calculate the start and end of each rank to extract data from the main image data.
  + We have three types of ranks :-
    - If first rank: we send to him its height and add to him the next width.
    - If last rank: we send to him its height and add to him the previous width.
    - Else ranks: we send to him its height and add to him the previous and next width.
  + The additional widths that send to other processors that we send is to untie the related data as we need for example for the last width for any rank will depend on the next width and so one for all ranks so we send additional widths to other processors but calculate the additional width in its rank.
* **Master Rank / Its role :-**
  + Rank 0.
  + Read the image data.
  + Send to other ranks there’s data using send.
  + Receive from other ranks the modified data.
  + Collect the received data in the main image data.
  + Calculate the time taken.
* **How to process on subarray that has been send to the ranks :-**
  + For every pixel we need to get its 8 neighbors in 3\*3 matrix.
  + Center, Left, right, up, down, left up, left down, right up, right down points.
  + If found the neighbor take it.
  + If not found the neighbor duplicate the pixel then take it.
  + Then we multiply the 3\*3 matrix with kernel matrix.
  + Put the result in the ansarray.
  + Hint: for every pixel we get its neighbors by using some equations.
* **Collect the processed data :-**
  + Every rank send the processed data to rank 0.
  + Rank 0 collect it according to the correct order of the data.
  + Put the collected data in the main image data.
* **Output with different number of processors :-**

