SANSKAAR PATNI
180905134
CSE-C 23
PPLAB ENDSEM

given : RES - o/p matrix NxN
A - i/p char matrix NxN

read N, A(each string ≤ N)
grid (2,2) - 2D grid
block (N/2, N/2) - 2D block
len stores count of characters in each row
10 strings!

```
%%cu
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#include<ctype.h>
__global__ void convertMatrix(char *A,
char *RES, int *len, int N){
    int Row=blockIdx.y*blockDim.y+threadIdx.y;
    int Col=blockIdx.x*blockDim.x+threadIdx.x;
  //will fill all corner edges(border elements)
  if(Row==0||Row==(N-1)||Col==0||Col==(N-1))
  {
      RES[Row*N+Col]='!';
  } //checks if unfilled
  else if(Col >= len[Row])){
      int primeRow = 0; /*flag*/
      for(int i=2; i<=primeRow/2; i++){
          if(Row% i==0){
              primeRow=1; //Row is not prime
              break;
          }
      }
~~[illegible]~~ // row number is prime or row=1
if ((primeRow == 0) || ~~[illegible]~~ (row == 1)){
        RES[row*N ~~[illegible]~~ + col] = '*';
}
else{
        RES[ROW*N + Col] = '#';
}
} // closes else if
else if( ~~[illegible]~~ A[ROW*N+Col] != '\0') {
    // filled elements
   // toggle only vowels.
    char ch= A[ROW*N + col] ;
   if( ch=='a' || cha=='e' || ch=='i' ||
     ch == 'o' || cha=='u'' || ch=='A' ||
        ch==E' || ch == 'I' || ch =='O' ||
        ch == 'U') {
        // if ch is lowercase convert to uppercase
        if (ch >= 'a' && ch <= 'z')
          RES[ROW*N+Col] = A[ROW*N + Col] - 32;
        else if(ch >= 'A' && ch <= 'Z')
          RES[ROW*N+ Col] = A[ROW*N + Col]+32;
   }
   else{
     // filled but not a vowel and not a border element
        RES[ROW*N + Col] = A[ROW*N + Col];
   }
} // closes kernel function
```

oh do we have to do this a sparse matrix?

↑ ~~did it like this~~

```
int main () {

    char **A, *RES, *dA, *dRES;
    int N;
    printf("Enter N(an even number):");
    scanf("%d", &N);
    int len[N]; int *d_len; char *str;
    for(int i=0; i<N; i++)
    {
                             str
     ~~pa~~ scanf("%s", ~~A[i]~~);
       len[i] = strlen(str); // store lenth of string in i th index
       A[i] = (char*)malloc(N*sizeof(char));
       strcpy(A[i], str);
    }

    int size1=N*N*sizeof(char);
    int size2=N*sizeof(int); // for d_len
// Allocate device memory
    cudaMalloc((void**)&dA, size1);
    cudaMalloc((void**)&dRES, size1);
    cudaMalloc((void**)&d_len, size2);
// copy from host to device
  cudaMemcpy(dA, A, size1, cudaMemcpyHostToDevice);
  cudaMemcpy(d_len, len, size2, cudaMemcpyHostToDevice);
dim3 dimBlock(ceil(N/2.0), ceil(N/2.0));
~~convertMatrix <<(2,2)~~e.
dim3 dimGrid(2, 2);
// 2D grid 2D block
convertMatrix<<<dimGrid, dimBlock>>>(dA,
 dRES, d_len, N);

cudaMemcpy(RES, dRES, size1, cudaMemcpyDeviceToHost);
```

Scanned with CamScanner

classmate
Date
Page

```
printf("Sample input RES:");
for(int i=0; i<N; i++)
{
  for(int j=0; j<N; j++)
    {
      printf("%c", RES[i][j]);
    }
  printf("\n");
}
cudaFree(d_A);
cudaFree(d_RES);
cudaFree(d_len);

 return 0;

} // host ~~code~~ code.
```

So basically solved it as if the char matrix was a sparse matrix kind, so I have used ~~[illegible]~~ len array which contains length of 10 arrays each.
A[i] allocated space length is N but actual length = strlen(str).
if element ~~[illegible]~~ Col >= len[Row] means it is aqunfilled element.

~~//Maybe should have asked~~
~~//Should have taken [illegible] of chars there a row pls.~~