2/6/20

Roll - 31

Rajar

Reg - 17090252

Name - Rajar Veeraraghavan

Roll - 31

Sem - VI

Section - D

---

1)

```
# include < mpi.h>
# include <stdio.h>

# define MCW MPI_COMM_WORLD

int main ( int argc, char const *argv[])
{
    int rank, n, a[4][4], res[4][4], csr[4], R[4], key, col[4];
    MPI_Status status;
    MPI_Init (&argc, &argv);
    MPI_Comm_rank ( &rank);
    n = 4;
    if (rank == 0)
    {
        for (int i=0; i<n; i++){
            for (int j=0; j<n; j++){
                scanf ("%d", &a[i][j]);
```

Roll -31
Rajan

```
    }
}

MPI_Scatter(a, 4, MPI_INT, arr, n, MPI_INT, 0, MCW);

MPI_Alltoall(arr, 1, MPI_INT, col, 1, MPI_INT, MCW);

MPI_Reduce(col, k, 4, MPI_INT, MPI_BAND, 0, MCW);

if (rank == 0)
{
    for(int i=0; i<3; i++)
    {
        MPI_Send(&k[i], 1, MPI_INT, i+1, 1, MCW);
    }
    key = k[3];
}
else {
    MPI_Recv(&key, 1, MPI_INT, 0, 1, MCW, &status);
}
for(int i=0; i<n; i++)
{
    arr[i] += key; }
```

```
MPI_Gather ( cor, n, MPI_INT, res, n, MPI_INT, 0, MCW);

if (rank == 0){

    for(int i=0; i<n; i++){
        for(int j=0; j<n; j++){
            printf ("%d ", res[i][j]);
        }
        printf ("\n");
    }
}

MPI_Finalize ();

return 0;

}
```

Roll - 31

Raja

2)

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

__global__ void func (char *S, char *A, int len)
{
    int row = blockIdx.x;
    int col = threadIdx.x;
    int maxLen = blockDim.x;
    int count = gridDim.x;
    int space = 0;
    int i=0; char c;
    while (i < len)
    {
        if (space == row) { break; }
        if (S[i] == ' ') { space++; }
        i++;
    }
    int j = i;
```

Roll -31

Rajat

```
while ( j < len)
{   if (S[j]==' ')
      {  break;}

    j++;
 }

 if ( col > j - i-1){

        A[maxlen * row + col] = '*' ;

 }
 else{
    c = S[i + col];

    if ( c='A' || c=='E' || c=='I' || c=='O' || c=='U')
    {   A[maxlen * row + col] = c+32 ;}

   else if ( c=='a' || c=='e' || c=='i' || c=='o' || c=='u')
    {   A[maxlen * row + col] = c - 32 ;}

   else {   A[maxlen * row + col] = c ;}
}
```

Roll - 31
Rajiv

```
}

int main()
{   char S[80];
    scanf("%s",
        gets(S);
        int len = strlen(S);
        int maxlen = 0;
        int wordCount = 1;
          int wordLen = 0;
        for(int i=0; i< len; i++){
              if (S[i] == ' '){  wordCount++;}
                  if ( wordLen > maxlen){
                          maxlen = wordLen;
                  }
                  wordLen=0;
             }
             else {  wordLen++;}
    }
```

Roll -31
Rajes

```
char A[wordCount][maxlen];

char * d_str, * d_A;

cudaMalloc( & d_str, len * sizeof (char));

cudaMalloc( & d_A, maxlen * wordCount * sizeof (char));

cudaMemcpy ( d_str, S, len * sizeof (char), cudaMemcpyHostToDevice);

func << wordCount, maxlen >> (d_str, d_A, len) ;

cudaMemcpy ( A, d_A, wordCount * maxlen * sizeof (char),
                cudaMemcpyDeviceToHost);

for (int i=0; i < WordCount; i++){
    for (int j=0; j < maxlen ; j++){
        printf ("%c ", A [i][j]);
    }
    printf (" \n");
}
}
```