```
!apt-get --purge remove cuda nvidia* libnvidia-*
!dpkg -l | grep cuda- | awk '{print $2}' | xargs -n1 dpkg --purge
!apt-get remove cuda-*
!apt autoremove
!apt-get update


!wget https://developer.nvidia.com/compute/cuda/9.2/Prod/local_installers/cuda-repo-ubuntu
!dpkg -i cuda-repo-ubuntu1604-9-2-local_9.2.88-1_amd64.deb
!apt-key add /var/cuda-repo-9-2-local/7fa2af80.pub
!apt-get update
!apt-get install cuda-9.2


!nvcc --version


!pip install git+git://github.com/andreinechaev/nvcc4jupyter.git


%load_ext nvcc_plugin
```

ARUNIMA SINGH THAKUR

SECTION C

ROLL NO. 31

PCAP LAB 7

28 MAY 2021

180905218

QUESTION 1

```
%%cu
#include <stdio.h>
#include <stdlib.h>
#include "cuda_runtime.h"
#include "device_launch_parameters.h"

__global__ void WordCount(char* str,int* d_count,int* len_words,char* word)
{

  int tempIdx=threadIdx.x;
  int idx=0;
  for(int i=0;i<tempIdx;i++)
  {
      idx += len_words[i]+1;
  }

  char s[10];
  int x=0;
```

```
    for(int j=0;j<len_words[tempIdx];j++){
        s[x++]=str[idx+j];
    }
    s[x] = '\0';

    int flag=0;
    for(int k=0;k<len_words[tempIdx];k++){
        if(s[k]!=word[k]){
            flag=1;
            break;
        }
    }
    if(flag==0){
        atomicAdd(d_count,1);
    }
}

int main(){
    char str[]="hello hello world hello";
    char word[]="hello";
    int num_words=1;
    int count=0;
    int result;
    int i=0;
    while(str[i]!='\0')
    {
        if(str[i]==' ')
        num_words++;
        i++;
    }
    int len_words[num_words];
    int len=0,j=0;
    i=0;
    while(str[i] != '\0')
    {
        len++;
        if(str[i]==' '){
            len--;
            len_words[j++]=len;
            len=0;
        }
        i++;
    }
    len_words[j]=len;

    int total_words=sizeof(len_words)/sizeof(int);

   char* d_str;
   int* d_count;
   char* d_word;
   int* d_len_words;

    //Allocate memory
    cudaMalloc((void**)&d_str,strlen(str)*sizeof(char));
```

```
  cudaMalloc((void**)&d_count,sizeof(int));
  cudaMalloc((void**)&d_word,strlen(word)*sizeof(char));
  cudaMalloc((void**)&d_len_words,total_words*sizeof(int));

  //Copy to device
  cudaMemcpy(d_str,str,strlen(str)*sizeof(char),cudaMemcpyHostToDevice);
  cudaMemcpy(d_count,&count,sizeof(int),cudaMemcpyHostToDevice);
  cudaMemcpy(d_word,word,strlen(word)*sizeof(char),cudaMemcpyHostToDevice);
  cudaMemcpy(d_len_words,len_words,total_words*sizeof(int),cudaMemcpyHostToDevice);


  //Launch Kernel

  WordCount<<<1,total_words>>>(d_str,d_count,d_len_words,d_word);

  //copy to host
  cudaMemcpy(&result,d_count,sizeof(int),cudaMemcpyDeviceToHost);

  //Check results
  printf("Total occurences of word %s is %d\n",word,result);

  cudaFree(d_str);
  cudaFree(d_count);
  cudaFree(d_word);
  cudaFree(d_len_words);


}
```

OUTPUT:

Total occurences of word hello is 3

QUESTION 2

```
%%cu
#include <stdio.h>
#include <stdlib.h>
#include "cuda_runtime.h"
#include "device_launch_parameters.h"
__global__ void wordcount_kernel(char* d_Sin, char* d_Sout, int len){
    int id=threadIdx.x;
    int s = (int)(id*(id-1)/2);
    if(s<0)
      s=0;
    int st = id*len - s;
    int l = len-id;
    for(int j=0;j<l;j++)
    {
        d_Sout[st+j] = d_Sin[j];
    }

}
```

```
}
int main() {
int len = 4;
char Sin[4]={'P', 'C', 'A', 'P'};
int outlen = (int)(len*(len+1)/2);
char Sout[outlen];
printf(" input String is %s\n", Sin);
char *d_Sin, *d_Sout;
cudaMalloc( (void **)(&d_Sin), sizeof(Sin));
cudaMalloc( (void **)(&d_Sout), sizeof(Sout));
cudaMemcpy(d_Sin,Sin,sizeof(Sin),cudaMemcpyHostToDevice);
wordcount_kernel<<<1,len>>>(d_Sin, d_Sout, len);
cudaError_t err = cudaMemcpy(Sout,d_Sout,sizeof(Sout),cudaMemcpyDeviceToHost);
if(err != cudaSuccess){
    printf("CUDA ERROR: %s\n", cudaGetErrorString(err));
}
printf("Result: %s\n", Sout);
cudaFree(d_Sin);
cudaFree(d_Sout);
return 0;
}
```

OUTPUT:

input String is PCAP

Result: PCAPPCAPCP

✓ 0s completed at 1:37 PM