WEEK 8 : String Programming in CUDA
Name : Sagnik Chatterjee
Reg: 180905478
Roll No: 61
Sec: VI B

Q2 Write a CUDA program that reads a string S and produces the string RS as follows:
Input string S: PCAP
Output string RS: PCAPPCAPCP
Note: Each work item copies required number of characters from S in RS.

=>

```
%%cu
#include <stdio.h>
#include <stdlib.h>

__global__ void patternPrint(char *k_input,char *k_output,int len){
   int id =threadIdx.x;
   int s1 =(int)(id*(id-1)/2);
   if(s1<0){
       s1=0;
   }
   int st = id*len - s1 ;
   int l = len - id;
   for(int i=0;i<l;i++){
       k_output[st+i] = k_input[i];
   }
}

int main(){
   int len=4;
   char input[4]={'P','C','A','P'};
   int outputlength =(int)(len*(len+1)/2);
   char output[outputlength];
```

```
   printf("Input string is :%s\n",input);
   char *k_input ,*k_output;

   cudaMalloc((void**)(&k_input),sizeof(input));
   cudaMalloc((void**)(&k_output),sizeof(output));
   cudaMemcpy(k_input,input,sizeof(input),cudaMemcpyHostToDevice);
   patternPrint<<<1,len>>>(k_input,k_output,len);
   cudaError t  =
cudaMemcpy(output,k_output,sizeof(output),cudaMemcpyDeviceToHost);
   if(t!= cudaSuccess){
       printf("Error : %s\n",cudaGetErrorString(t));
   }
   printf("result : %s\n",output);
   cudaFree(k_input);
   cudaFree(k_output);

   return 0;
}
```

Screenshot:

```
Input string is :PCAP
result : PCAPPCAPCP
```

Q1 Write a program in CUDA to count the number of times a given word is repeated in a sentence.
(Use Atomic function)

=> Code

```
%%cu
#include <cstdio>
#include <math.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
```

```
__global__ void findWordCount(char* str, char* word, int* count, int strl,
int wordl)
{
   //here  every thread  will skip the previous words and check for only
its words
 // for that thread only
   int id = threadIdx.x;
   int c = 0;
   int i = 0;
   for (i = 0; i < strl; i++)
   {
       if (str[i] == ' ')
           c++;
       if (c == id)
       {
           break;
       }
   }
   char* w = (char*)malloc(sizeof(char) * 30);
   int j = 0;
   if (i != 0)
       i = i + 1;
   for (;i < strl;i++)
   {
       if (str[i] == ' ')
           break;
       w[j++] = str[i];
   }
   w[j] = '\0';
   printf("Current Thread --> %d\n , and the current word checking is
%s\n",id,w);

   int flag = 0;
   for (i = 0;i < j;i++)
       if (word[i] != w[i])
       {
           flag = 1;
          break;
       }
```

```
    if (flag == 0){
        atomicAdd(count, 1);
    }
}

int main()
{
    char* string1 = (char*)malloc(sizeof(char) * 300);
    char* pattern = (char*)malloc(sizeof(char) * 30);
    string1 = "LAMBDA CALCULAS IS DEAD LONG LIVE LAMBDA";
    pattern = "LAMBDA";
    int* wordCount = (int*)malloc(sizeof(int));
    *wordCount = 0;
    printf("Given String: %s\n ",string1);
    printf("Given Word: %s\n", pattern);

    // Count the number of words in the string

    int count = 1; //For the last word
    for (int i = 0;i < strlen(string1);i++)
    {
        if (string1[i] == ' ')
            count++;
    }
    char* d_str, * d_word;
    int* d_wordCount;

    cudaMalloc((void**)&d_str, sizeof(char) * 300);
    cudaMalloc((void**)&d_word, sizeof(char) * 30);
    cudaMalloc((void**)&d_wordCount, sizeof(int));

    cudaMemcpy(d_str, string1, sizeof(char) * 300, cudaMemcpyHostToDevice);
    cudaMemcpy(d_word, pattern, sizeof(char) * 30, cudaMemcpyHostToDevice);
    cudaMemcpy(d_wordCount, wordCount, sizeof(int),
cudaMemcpyHostToDevice);

    dim3 gridDim(1, 1, 1);
    dim3 blockDim(count, 1, 1);
```

```
   findWordCount <<< 1, count >>> (d_str, d_word, d_wordCount,
strlen(string1), strlen(pattern));
   cudaMemcpy(wordCount, d_wordCount, sizeof(int),
cudaMemcpyDeviceToHost);
   printf("Total count of %s in %s is : %d\n", pattern,
string1,*wordCount);

   cudaFree(d_word);
   cudaFree(d_wordCount);
   cudaFree(d_str);



   return 0;
}
```

Output:

```
Given String: LAMBDA CALCULAS IS DEAD LONG LIVE LAMBDA
 Given Word: LAMBDA
Current Thread --> 0
 , and the current word checking is LAMBDA
Current Thread --> 1
 , and the current word checking is CALCULAS
Current Thread --> 2
 , and the current word checking is IS
Current Thread --> 3
 , and the current word checking is DEAD
Current Thread --> 4
 , and the current word checking is LONG
Current Thread --> 5
 , and the current word checking is LIVE
Current Thread --> 6
 , and the current word checking is LAMBDA
Total count of LAMBDA in LAMBDA CALCULAS IS DEAD LONG LIVE LAMBDA is : 2
```