PP Lab 7 : CUDA-3

Name: Rajvardhan Reddy

Registration Number: 180905093

Section: B

Batch: B1

Roll Number: 19

P1) Write a program in CUDA to count the number of times a given word is repeated in a sentence (Use Atomic function) .

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

__global__
void word_count_kernel(char* str, char* key, int* word_indices, int* result)
{
   int idx = threadIdx.x + blockIdx.x * blockDim.x;

   // get idx'th word
   int si = word_indices[idx];
   int ei = word_indices[idx+1];
```

```
char word[100];

int i = 0;
for (i = 0; i < (ei-si-1); i++)
{
   word[i] = str[si+1+i];
}
word[i] = '\0';

// compare word and key
int i1 = 0;
int i2 = 0;
int is_equal = 1;

while (word[i1] != '\0' && key[i2] != '\0')
{
   if (word[i1] == key[i2])
   {
      i1++;
      i2++;
   }
   else
   {
      is_equal = 0;
      break;
   }
```

```
    }

    if (is_equal == 1)
    {
        atomicAdd(result, 1);
    }
}

int main()
{
    char str[100] = " apple banana mango apple laptop apple ";
    char key[100] = "apple";

    int str_len = strlen(str);
    int key_len = strlen(key);

    int word_count = 0;

    for (int i = 0; i < str_len; i++)
    {
        if (str[i] == ' ')
        {
            word_count++;
        }
    }
    word_count--;
```

```
    int* word_indices = (int*) (malloc(word_count * sizeof(int)));
    int wi = -1;

    for (int i = 0; i < str_len; i++)
    {
        if (str[i] == ' ')
        {
            word_indices[++wi] = i;
        }
    }

    int result = 0;

    char* d_str;
    char* d_key;
    int* d_word_indices;
    int* d_result;

    cudaMalloc((void**)&d_str, str_len * sizeof(char));
    cudaMalloc((void**)&d_key, key_len * sizeof(char));
    cudaMalloc((void**)&d_word_indices, (word_count+1) * sizeof(int));
    cudaMalloc((void**)&d_result, sizeof(int));

    cudaMemcpy(d_str, str, str_len * sizeof(char), cudaMemcpyHostToDevice);
    cudaMemcpy(d_key, key, key_len * sizeof(char),
cudaMemcpyHostToDevice);
```

```
    cudaMemcpy(d_word_indices, word_indices, (word_count+1) * sizeof(int), cudaMemcpyHostToDevice);
    cudaMemcpy(d_result, &result, sizeof(int), cudaMemcpyHostToDevice);

    word_count_kernel<<<1, word_count>>>(d_str, d_key, d_word_indices, d_result);

    cudaMemcpy(&result, d_result, sizeof(int), cudaMemcpyDeviceToHost);

    printf("Input String: %s\n", str);
    printf("Key: %s\n", key);
    printf("Total occurances of %s is %d\n", key, result);

    cudaFree(d_str);
    cudaFree(d_key);
    cudaFree(d_result);

    return 0;
}
```

Output:

```
Input String:  apple banana mango apple laptop apple
Key: apple
Total occurances of apple is 3
```

P2) Write a CUDA program that reads a string *Sin* and produces the string *Sout* as follows:

Input string *Sin*: PCAP Output string *Sout*: PCAPPCAPCP

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

__global__
void kernel(char* sin, int* sin_len, char* sout)
{
    int idx = threadIdx.x + blockIdx.x * blockDim.x;

    int si = 0; // start index
    for (int i = 0; i < idx; i++)
    {
        si += (*sin_len)-i;
    }

    int total_chars = (*sin_len) - idx;
    for (int i = 0; i < total_chars; i++)
```

```
    {
        sout[si++] = sin[i];
    }
}

int main()
{
    char sin[100] = "PCAP";
    char sout[100];
    int sin_len = strlen(sin);
    int sout_len = 0;
   for (int i = 0; i < sin_len; i++)
    {
        sout_len += (i+1);
    }
    char* d_sin;
    int* d_sin_len;
    char* d_sout;

    cudaMalloc((void**) &d_sin, sin_len * sizeof(char));
    cudaMalloc((void**) &d_sin_len, sizeof(int));
```

```
    cudaMalloc((void**) &d_sout, (sout_len + 1) * sizeof(char));


    cudaMemcpy(d_sin, sin, sin_len * sizeof(char),
cudaMemcpyHostToDevice);

    cudaMemcpy(d_sin_len, &sin_len, sizeof(int),
cudaMemcpyHostToDevice);

    cudaMemcpy(d_sout, sout, (sout_len + 1) * sizeof(char),
cudaMemcpyHostToDevice);


    kernel<<<1, sin_len>>>(d_sin, d_sin_len, d_sout);


    cudaMemcpy(sout, d_sout, (sout_len + 1) * sizeof(char),
cudaMemcpyDeviceToHost);


    sout[sout_len] = '\0';


    printf("Sin: %s\n", sin);

    printf("Sout: %s\n", sout);

    cudaFree(d_sin);

    cudaFree(d_sin_len);

    cudaFree(d_sout);

     return 0;

}
```

Output :