PP LAB WEEK-9

DSE VI-A2 Divansh Prasad 210968140

1) Write a program in CUDA to count the number of times a given word is repeated in a sentence.

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

#define MAX_SENTENCE_LENGTH 1000
#define MAX_WORD_LENGTH 100

__global__ void countWordOccurrences(char *sentence, char *word, int
*count, int sentenceLength, int wordLength) {
   int tid = blockIdx.x * blockDim.x + threadIdx.x;
   int wordCount = 0;
   if (tid < sentenceLength - wordLength + 1) {
       int match = 1;
       for (int i = 0; i < wordLength; ++i) {
           if (sentence[tid + i] != word[i]) {
               match = 0;
               break;
           }
       }
       if (match) {
           atomicAdd(count, 1);
       }
   }
}

int main() {
   char sentence[MAX_SENTENCE_LENGTH];
   char word[MAX_WORD_LENGTH];
   int sentenceLength, wordLength;

   printf("Enter a sentence: ");
   fgets(sentence, MAX_SENTENCE_LENGTH, stdin);
```

```
    sentenceLength = strlen(sentence);
    if (sentence[sentenceLength - 1] == '\n') {
        sentence[sentenceLength - 1] = '\0';
        sentenceLength--;
    }

    printf("Enter the word to count: ");
    scanf("%s", word);
    wordLength = strlen(word);
    char *d_sentence, *d_word;
    int *d_count;
    int count = 0;
    cudaMalloc(&d_sentence, sentenceLength * sizeof(char));
    cudaMalloc(&d_word, wordLength * sizeof(char));
    cudaMalloc(&d_count, sizeof(int));
    cudaMemcpy(d_sentence, sentence, sentenceLength * sizeof(char),
cudaMemcpyHostToDevice);
    cudaMemcpy(d_word, word, wordLength * sizeof(char),
cudaMemcpyHostToDevice);
    cudaMemcpy(d_count, &count, sizeof(int), cudaMemcpyHostToDevice);
    int threadsPerBlock = 256;
    int blocksPerGrid = (sentenceLength + threadsPerBlock - 1) /
threadsPerBlock;
    countWordOccurrences<<<blocksPerGrid, threadsPerBlock>>>(d_sentence,
d_word, d_count, sentenceLength, wordLength);
    cudaMemcpy(&count, d_count, sizeof(int), cudaMemcpyDeviceToHost);
    printf("Number of occurrences of '%s' in the sentence: %d\n", word,
count);
    cudaFree(d_sentence);
    cudaFree(d_word);
    cudaFree(d_count);
    return 0;
}
```

```
divansh@ROG-STRIX:~/Desktop/PP-Lab/Week-9$ nvcc -o WordCount WordCount.cu
WordCount.cu(10): warning #177-D: variable "wordCount" was declared but never referenced

divansh@ROG-STRIX:~/Desktop/PP-Lab/Week-9$ ./WordCount
Enter a sentence: Hello, World! CUDA: CUDA says Hello, World!
Enter the word to count: CUDA
Number of occurrences of 'CUDA' in the sentence: 2
divansh@ROG-STRIX:~/Desktop/PP-Lab/Week-9$ ./WordCount
Enter a sentence: x x x x x x x x y y y y y y x x x x x y y y y
Enter the word to count: y
Number of occurrences of 'y' in the sentence: 10
```

2) Write a CUDA program that reads a string S and produces the string RS as follows:
Input String S: PCAP
Output String RS: PCAPPCAPPCP
Note: Each work item copies the required number of characters from S to RS.

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

#define MAX_STRING_LENGTH 1000

__global__ void repeatString(char *S, char *RS, int length, int S_length)
{
    int tid = blockIdx.x * blockDim.x + threadIdx.x;
    if (tid < length) {
        RS[tid] = S[tid % S_length];
    }
}

int main() {
    char S[MAX_STRING_LENGTH];
    char RS[MAX_STRING_LENGTH * 3]; // Assuming the output string can be at
most 3 times the length of the input string
    int length;

    printf("Enter a string: ");
    fgets(S, MAX_STRING_LENGTH, stdin);
    length = strlen(S);
    if (S[length - 1] == '\n') {
        S[length - 1] = '\0';
```

```
        length--;
    }

    int S_length = length; // Store the length of input string
    length *= 3; // Adjust length for output string

    char *d_S, *d_RS;
    cudaMalloc(&d_S, S_length * sizeof(char));
    cudaMalloc(&d_RS, length * sizeof(char));

    cudaMemcpy(d_S, S, S_length * sizeof(char), cudaMemcpyHostToDevice);

    int threadsPerBlock = 256;
    int blocksPerGrid = (length + threadsPerBlock - 1) / threadsPerBlock;

    repeatString<<<blocksPerGrid, threadsPerBlock>>>(d_S, d_RS, length,
S_length);

    cudaMemcpy(RS, d_RS, length * sizeof(char), cudaMemcpyDeviceToHost);

    printf("Input String S: %s\n", S);
    printf("Output String RS: %s\n", RS);

    cudaFree(d_S);
    cudaFree(d_RS);

    return 0;
}
```

```
divansh@ROG-STRIX:~/Desktop/PP-Lab/Week-9$ nvcc -o S-to-RS S-to-RS.cu
divansh@ROG-STRIX:~/Desktop/PP-Lab/Week-9$ ./S-to-RS
Enter a string: PCAP
Input String S: PCAP
Output String RS: PCAPPCAPPCAP
divansh@ROG-STRIX:~/Desktop/PP-Lab/Week-9$ ./S-to-RS
Enter a string: xyz
Input String S: xyz
Output String RS: xyzxyzxyzh!
divansh@ROG-STRIX:~/Desktop/PP-Lab/Week-9$ ./S-to-RS
Enter a string: a
Input String S: a
Output String RS: aaa
```