(a)

numblocks	blocksize	time_for_kernel_kmerPosConcat	time_for_kernel_kmerOffsetFill	time_for_kernel_kmerPosMask
64	32	2.6800ms	1.4497ms	152.60us
64	128	834.55us	886.61us	39.840us
64	512	294.43us	934.77us	12.192us
256	32	836.82us	900.50us	41.344us
256	128	286.21us	883.58us	12.640us
256	512	310.85us	3.4537ms	9.6320us
1024	32	276.61us	904.80us	12.704us
1024	128	312.42us	3.6546ms	9.6320us
1024	512	266.91us	1.8759ms	12.928us

(b)
For me the best speed up is achieved when numblocks = 256 and block size = 128 are set.

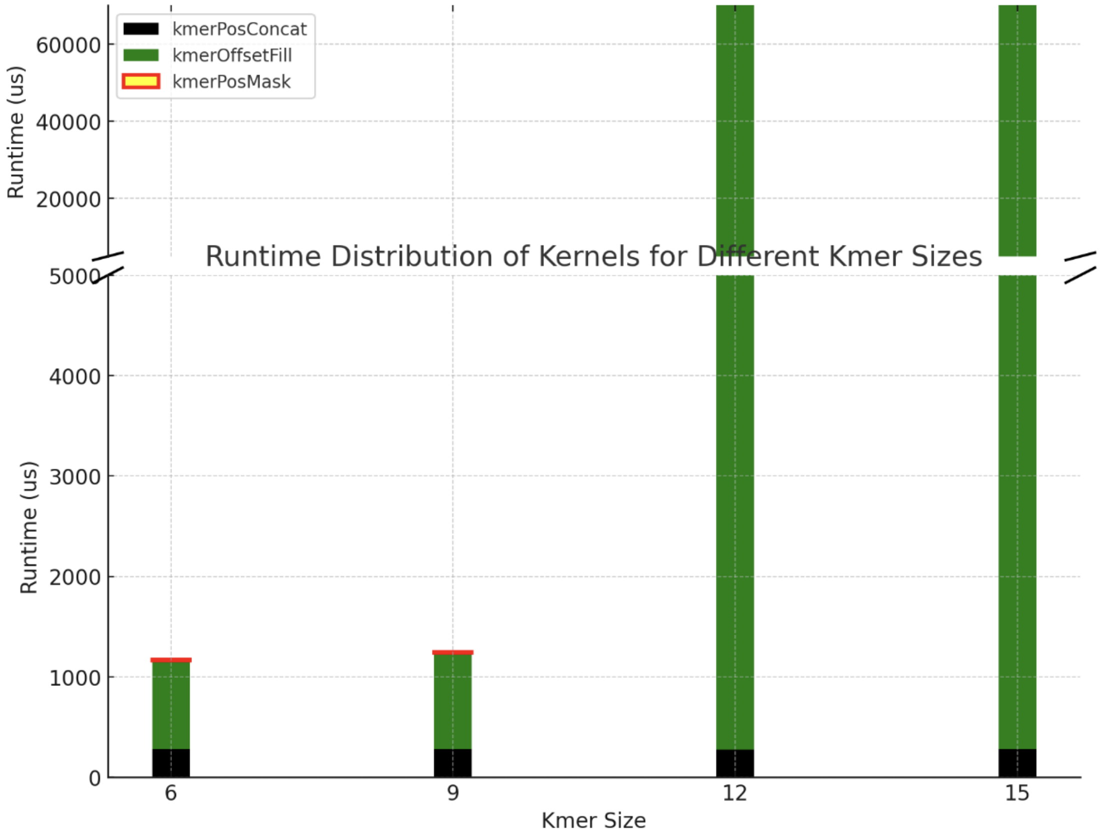
(c)
Speedup for best case over sequential exec:

Speed up for kmerPosConcat	Speed up for kmerOffsetFill	Speed up for kmerPosMask
2486	1543	78498

(d)
cudaMalloc: **116.32ms**: Approx 25% of kernel execution time
[CUDA memcpy HtoD]: **621.44us**: Approx equal to kernel execution time
[CUDA memcpy DtoH]: **3.4880us**: Very fast compared to kernel execution time
cudaMemcpy: **850.75us**: Approx equal to kernel execution time
cudaFree: **8.4476ms**: Very high compared to kernel execution time.

(e)

Kmersize	kmerPosConcat	kmerOffsetFill	kmerPosMask
6	284.00us	877.18us	12.352us
9	280.96us	955.61us	12.832us
12	277.38us	3.7805ms	12.480us
15	282.30us	64.832ms	13.087us



Bonus (a):

kmerOffset array vary with exponentially with kmerSize (4**kmersize)

Can a seedTable constructed for a fixed kmerSize be used to search for k-mer occurrences larger than the kmerSize? If yes, how? If no, why not? ANS : No

A k-mer is a substring of text of length k. Therefore, a seed table built for a specific kmerSize will have entries that correspond to all the possible k-mers of that exact length

Suggest some ways to construct a seed lookup table under reasonable memory for very large kmerSize (e.g. kmerSize > 40).

Compression can be achieved by grouping 2 similar bases and processing only subsets of the seed table and storing these subsets in the form of mappings. Sort of virtual memory analogy.