Problem I

The assembly code generated for the compiled loop of combine3 with optimization level 2 is shown below:

combine3: data_t = float, OP = *, compiled –O2

in %rdx, data in %rax, limit in %rbp, dest at %rx12, product in %xmm0

```
1 .L560:                          loop:
2 mulss   (%rax, %rdx, 4), %xmm0  Multiply product by data[i]
3 addq    $1, %rdx                Increment i
4 cmpq    %rdx, %rbp              Compare limit : i
5 movss   %xmm0, (%r12)           Store product at dest
6 jg      .L560                   If >, go loop
```

Illustrate the code above with **data-flow graph** like figure 5.14(a) or (b) in textbook and mark the **critical path**. You can use "store" to identify operation in line 5

Problem II

The following code for PrefixSum seems not very good. Please rewrite it .

1) Please not repeatedly retrieve the value of p[i] from memory.
2) Please use **three-way loop unrolling** to reduce the operations in each loop.
3) Please use **three-way parallelism.**

```
/*Compute prefix sum of vector v*/
1    void prefixSum(float v[], float p[], long int n)
2    {
3        long int i
4        p[0] = v[0];
5        for ( i = 1; i < n; i++ )
6            p[i] = p[i-1] + v[i];
7    }
```

Problem III

Suppose an allocator has a heap with size of **(5 * SIZE)**, and the heap is **empty** initially. Then, a user makes a sequence of requests as shown in the table below.

a). What's the aggregate payload after the execution of code in line 7? What's the peak memory utilization after these requests?

```
1.  void *pa, *pb, *pc;
2.
3.  pa = malloc(SIZE);
4.  free(pa)
5.  pa = malloc(SIZE);
6.  pb = malloc(SIZE);
7.  free(pa)
8.  pa = malloc(SIZE);
9.  pc = malloc(SIZE);
10. free(pa);
11. free(pb);
12. free(pc);
```

b). How do we know how much memory to free just given a pointer?