1. **Struct iteration and Cache behavior**

Consider the following two scenarios. Which has a higher cache miss rate? Why might we choose to do either implementation? Assume that N >> cache_size and we have 64-byte cache lines.

Method 1:

```
struct airplane{
      int id_number;
      int num_passengers;
      float airport_x, airport_y;
      float coord_x, coord_y;
      float dir_x, dir_y;
      float vel_x, vel_y;
};
struct airplane airplanes[N];
float avg_dist_from_port = 0;
for(int i = 0; i < N; i++){
      float airport_x = airplanes[i].airport_x;
      float airport_y = airplanes[i].airport_y;
      avg_dist_from_port += sqrt((airport_x) ** 2 + (airport_y) **
2);
}
avg_dist /= N;
```

Method 2:

```
struct pair{
      float x,y;
}
int id_numbers[N];
int num_passengers[N];
struct pair airport_coords[N];
struct pair coords[N];
struct pair dir[N];
struct pair vel[N];

float avg_dist_from_port = 0;
for(int i = 0; i < N; i++){
      float airport_x = airport_coords[i].x;
      float airport_y = airport_coords[i].y;
      avg_dist_from_port += sqrt((airport_x) ** 2 + (airport_y) **
2);

}
avg_dist /= N;
```

2. **Critical Path**

Examine the code below that stores data into an accumulator from operations on elements of the vector *v*

```
void combine7(vec_ptr v, data_t *dest){
      long i;
      long length = vec_length(v);
      long limit = length-1;
      data_t *data = get_vec_start(v);
      data_t acc = IDENT:
      // combines 2 elements at a time
      for(i = 0; i < limit; i+=2){
            acc = acc OP (data[i] OP data[i+1]);
      }
      // finish any remaining elements
      for(; i < length; i++){
            acc = acc OP data[i];
      }
      *dest = acc;
}
```

a) For one iteration of the loop, what would the data-flow graph look like? In addition, highlight the critical path in the data flow.

b) When this function is given a vector size of *n,* what would the data-flow graph look like? Similar to part a, highlight the critical path in the entire data flow.

c) The function incorporates 2x1 loop unrolling, but what other ways can we modify the function to reduce the latency bound by half?

3. 1D and 2D Stencil optimization

In the performance lab, you will be optimizing some code for 3D stencil computation. In this question, let's take a look at 1D and 2D stencil computations. (A side note: in the performance lab and in this question, we simplified the notation of the stencil computation (for handling edge cases), so it may differ slightly from other sources you might find.)

(1) 1D Stencil

Assume `OUT_LEN` is N >> 10000, `KERN_LEN` is 2, and `IN_LEN` is N+2.

(a) Assume cache has 32B/line, what is the miss rate of `vanilla_1D_stencil`?

```
void vanilla_1D_stencil(double In[IN_LEN], double Out[OUT_LEN],
                        double Kern[KERN_LEN]) {
    for (int i = 0; i < OUT_LEN; i++) {
        for (int x = 0; x < KERN_LEN; x++) {
            Out[i] += In[i+x] * Kern[x];
    }   }
}
```

(b) Given the initialization below, what does the resulting `Out` array look like after calling `vanilla_1D_stencil`?

```
double* In = malloc(IN_LEN * sizeof(double));
for (int i = 0; i < IN_LEN; i++) { In[i] = i + 1;}

double* Kern = malloc(KERN_LEN * sizeof(double));
for (int i = 0; i < KERN_LEN; i++) { Kern[i] = 1.0/KERN_LEN;}

double* Out = malloc(OUT_LEN * sizeof(double));
for (int i = 0; i < OUT_LEN; i++) {Out[i] = 0;}
```

(2) 2D Stencil

Assume `OUT_LEN` is N >> 10000, `KERN_LEN` is 4, and `IN_LEN` is N+4. Assume the cache has 32B/line, and the total cache size is smaller than 8N bytes.

(a) What is the miss rate of `vanilla_2D_stencil`?

```
void vanilla_2D_stencil(double In[IN_LEN][IN_LEN],
                        double Out[OUT_LEN][OUT_LEN],
                        double Kern[KERN_LEN][KERN_LEN]) {
    for (int i = 0; i < OUT_LEN; i++) {
        for (int j = 0; j < OUT_LEN; j++) {
            for (int x = 0; x < KERN_LEN; x++) {
                for (int y = 0; y < KERN_LEN; y++) {
                    Out[i][j] += In[i+x][j+y] * Kern[x][y];
    }   }   }   }
}
```

(b) How can we reduce the miss rate?