1. If we are using external merge sort we use the formula we learned in class. Which is P = 1 + Log(B-1) (N/B) which is 1 + Log(5) (1000000/6) = 9. So it would take approximately 9 passes.

2. It will go through 2 pointers. If you start at the root node you go down to the leaf node containing keys less than 10, then you follow another pointer to the next leaf node that contains keys less than 20. You can stop there since we are just looking for keys that are less than 19. So 2 pointers total.

3. The answer to this question would be 23, 23 in binary is 10111. So using either of the hash functions will sort it to the last bucket which already has 4 keys in it so it would cause an overflow.

4. For a sparse b+tree there will be a total of 19 nodes, with 10 leaf nodes each holding 2 values, then there are 5 intermediate and 3 nodes above that finally connected to the singular root node. So 19 total

5. Plan 2 is likely to be more efficient because the filter is applied before the join so you are just joining the tuples that you need to and not all of them.

6. This is not true because multi-threading is not the only way to get concurrent processing. You could also use task parallelism to achieve your goal.

7. You can implement a probe or bloom filter to optimize the join. You could also use the partitioned hash join, also called the Grace Hash Join. Make sure you use a highly useful hash function and have a plan for collisions.

8. This plan would use a total of 119 I/Os. In the beginning, scanning the applicants tables takes 100 I/Os and scanning the school table takes 10 I/Os. This will output 109 tuples and a sort-merge doesnt add to the cost and outputs 9 tuples. The index-nested loop takes an additional 9 I/Os. The final steps do not take any additional I/Os so 119 is the total.

9.
    a. To fix this you could use extendible hashing. This way when a bucket overflows you can update the hash function and rehash keys to make the distribution of keys more even.
    b. The formula to calculate this is M + [M / (B-2)] * N, so 2400 + (2400/73) * 1200. The answer is 42,000 I/Os

10. Given the amount of internal nodes is 2n then the number of leaf nodes would be 2n + 1, so to get an exact amount we would need the n value.

11. Since you try to insert into table 1 first then you would try table 2 the answer is A, since it would insert 12 no problem into table one and then it would try to insert 13 into table 1 and it would get a collision so it would try table 2 next.