Numbers (Spoiler)

For the first subtask, one only needs to bruteforce the solutions from a to b. For each number we check whether it is palindrom free and if it is, we just add one to a counter variable. The checking can be done quite easily in quadratic time depending on the number of digits by trying all the mid position of of possible palindromes. This gives us a solution in $O(18^2 * (a - b))$.

For the full-score solution, the key observation is that if a number contains a palindrom of length k (k>=3) it contains a palindrome of length two or three (just take the middle of the found palindrome and depending on whether it is odd or not, you take the middle and the left and right of the middle). For example,12323 contains 232 and 1221 contains 22. With this information, it is quite easy to do dynamic programming by saving the last two position, the length of the number, in order to get the number of palindrome-free numbers up to a certain number. See the source code in C++ for details.

A common mistake was to forget to consider 0 and to use long longs.

Source Code

See on the back.

```
#define ll long long
ll dp[10][10][20][2];
string str;

// all : all digits are allowed
ll Calc(int first, int second, int len, bool all){
        if(len>=str.length()){
                return 1;
        }else{
                if(dp[first][second][len][all]==-1){
                        ll help= 0;

                        if(all){
                                for(int n=0;n<=9;n++){
                                        if(n!=first && n!=second){
                                                help+=Calc(second,n, len+1, true);
                                        }
                                }
                        }else{
                                int limit = str[len]-'0';
                                for(int n=0;n<limit;n++){
                                        if(n!=first && n!=second){
                                                help+=Calc(second, n, len+1,true);
                                        }
                                }
                                if(limit!=first && limit!=second){
                                        help+=Calc(second, limit, len+1,false);
                                }
                        }
                        dp[first][second][len][all]=help;
                }
                return dp[first][second][len][all];
        }
}

ll Get(ll num){
                if(num<0) return 0;

                stringstream ss; ss << num; str = ss.str();

                memset(dp,-1,sizeof(dp));

                int first = str[0]-'0';
                ll res =1;

                for(int n=1;n<=first;n++){
                        res+=Calc(n,n,1,n!=first);
                }

                for(int k=2;k<=str.length();k++){
                        for(int n=1;n<10;n++){
                                res+=Calc(n,n,k,true);
                        }
                }

                return res;
}
int main(){
        ll a,b;
        scanf("%lld %lld",&a,&b);
        ll counter = Get(b)-Get(a-1);
        printf("%lld\n",counter);
}
```