1. brute.cc:

For this part, I start by creating an empty string, and then I append “a” n times where n is the length of the word. After that, I start by encrypting the string that I have now (eg. “aaaaa”) and then check to see if the resulting encryption is equal to the encryption (brute), then I would print the word. I used switch statements for the incrementing and an int carry to handle the case where I had to increment another column of the letters (eg. “aaaa5” to “aaaba”).

char	brute
4	46s
5	1min 24s
6	1min 55s

2. symbol.cc:

For this part, I start with a test and two empty strings firstHalf and secondHalf, and I start by setting the test string to “aaaaa” for example, and then generating a symbol table for N/2 letters where N is the number of letters in test. A while loop will generate a symbol table for each iteration of test and do comparisons until our key is found. I cushioned the unwanted bits of the second and first halves with a’s since their bit value is 0. I used a Boolean to act as a flip switch to know when we need a new symbol table.

char	symbol
5	57 sec
6	1min 47s
8	-
10	-

3. unique.cc:

I did not implement this part.