Problem 3 (25 points): C Strings

The C function below is intended to check whether string `s` contains the reverse of string `t`. For example, "`olleh`" is the reverse of "`hello`". If the two strings are the same when one is reversed, the function returns 1. Otherwise, it returns 0.

Unfortunately, Prof. Lumetta left a few bits out.

```
int backwards (char* s, char* t)
{
    char* u = s;

    while ('\0' != *u) { _______________ ; }       // blank #1

    while ('\0' != _______________ ) {             // blank #2

        if (s == _______________ || *u != *t) {    // blank #3

            ____________________ ;                 // blank #4

        }

        t++;

    }

    return (____________________);                 // blank #5
}
```

Circle EXACTLY ONE ANSWER to indicate what should appear in each blank in the code above.

1. (5 points) blank #1

A) `*u-- = *++s` B) `s++` C) `++t` D) `u++` E) `--u`

2. (5 points) blank #2

A) `*t++` B) `*++t` C) `*t` D) `*s` E) `*u`

3. (5 points) blank #3

A) `--u` B) `t--` C) `u - 1` D) `u--` E) `--t`

4. (5 points) blank #4

A) `*u = '\0'` B) `*u = *t` C) `--s` D) `return 0` E) `return 1`

5. (5 points) blank #5

A) `u - s` B) `u == t` C) `s == u` D) `'\0' == *s` E) `s++`

Problem 4 (25 points): Arrays

The function below is supposed to remove all negative numbers from an array. The function takes three parameters: `list` (the array of integers), `length` (the length of the array), and `sum_ptr` (a parameter written by `remove_neg` with the sum of all values removed from the array). The function returns the adjusted length of the array (the original length minus the number of negative numbers removed). You may assume that `length` is positive.

Unfortunately, Prof. Lumetta has again left blanks in the code.

Fill in the blanks to complete the code. Note that this problem is NOT MULTIPLE CHOICE. **Write the correct code directly on the blanks to complete the function.**

```
int
remove_neg ( _____________________________ ,                  // blank #1

             int length, int* sum_ptr)
{
    int ___________________________________ ;                  // blank #2

    int i;

    for (i = 0, *sum_ptr = 0; length > i; i++) {

        if ( _____________________________ ) {                 // blank #3

            list[adj_len++] = list[i];

        } else {

            *sum_ptr = _____________________________ ;         // blank #4

        }

    }

    ___________________________________ ;                      // blank #5

}
```