3) (10 pts) ALG (Stacks) Suppose we pass the string "cupcake" to the following function. What will the function's output be, and what will the stacks *s1* and *s2* look like when the function terminates? You may assume the stack functions are written correctly and that the stacks are designed for holding characters.

```
void string_shenanigans(char *str)
{
   int i, len = strlen(str);
   char *new_string = malloc(sizeof(char) * (len + 1));
   Stack s1, s2;
   init(&s1);
   init(&s2);

   for (i = 0; i < len; i++) {
      push(&s1, str[i]);  // this pushes onto stack s1
      push(&s2, str[i]);  // this pushes onto stack s2
   }

   for (i = 0; i < len; i++) {
      if (i % 2 == 0) {
         // Note: pop() returns the character being removed from the stack.
         if (!isEmpty(&s1))
            new_string[i] = pop(&s1);
         if (!isEmpty(&s1))
            push(&s2, pop(&s1));
      }
      else {
         pop(&s2);
         new_string[i] = pop(&s2);
      }
   }

   new_string[len] = '\0';
   printf("%s\n", new_string);
   free(new_string);
}
```

eeakpac	**(the stack is empty)**	**c <- top** **p** **u** **c**
printf() output	final contents of *s1* (please label 'top' for clarity)	final contents of *s2* (please label 'top' for clarity)

Grading: Award 7 points for the correct output ("eeakpac"), 1 pt per letter, 1 pt for the correct contents of *s1*, and 2 pts for the correct contents of *s2*. (Give 1 pt if the stack is flipped.) If the output looks reasonably close and has a minor error, feel free to award partial credit.