```
/*
 * strings
 */

/*
 * Synopsis: strings < unix_PS_file.images_split >
 *                     unix_PS_file.images_split.strings_isolated
 */

/*
 * Each / for defining a PostScript variable name is separated from its
 * predecessor by some white space or line break.
 *
 * Each string, which is a string of characters surround by ( and )
 * and in which an interior ) is escaped with a \, is made to begin on a
 * new line and if it is longer than 72 characters, the text following the
 * closed parenthesis at the end of the string also is made to begin on a
 * new line.
 *
 * Finally, except inside a string, a space or tab that occurs past
 * position 72 of an output line is converted into a ^J to help break the
 * text into editable lines.
 */

#include <stdio.h>
#define LIMIT 72

int i;  /* position of current character in output line */
char curr, prev;
void in_string_loop();

main()
{
        /* look through non-string characters */
        prev = '\n';
        i = 0;
        curr = getchar();

        while (curr != EOF) {

                switch (curr) {

                        /* put a space before a / that is not preceeded by
                        a NL, space, or tab */
                        case('/'): {
                                if (prev == '\n' ||
                                    prev == ' '  ||
                                    prev == '\t') {
                                        printf("%c", curr);
                                        i += 1;
                                } else { /* prev is not NL or WS */
                                        printf(" %c", curr);
                                        i += 2;
                                }
                                break;
                        }

                        /* put a NL before a string that is not preceeded by
                        a NL and then deal with the inside of a string */
                        case('('): {
                                if (prev == '\n') {
                                        printf("(" /*, curr */);
                                        i += 1;
                                } else { /* prev is WS or other */
                                        printf("\n%c", curr);
                                        i = 1;
```

```
				}
				in_string_loop();
				break;
			}

			/* if the current space or tab is after position LIMIT
			of the current output line, change it to a NL */
			case(’ ’):
			case(’\t’): {
				if (i < LIMIT) {
					printf("%c", curr);
					i += 1;
				} else {
					printf("\n");
					i = 0;
				}
				break;
			}

			/* after a NL, reset the position to 0 */
			case(’\n’): {
				printf("\n");
				i=0;
				break;
			}

			/* otherwise print the character and bump up the
			position */
			default: {
				printf("%c", curr);
				i += 1;
				break;
			}
		} /* end switch */

		prev = curr;
		curr = getchar();
	} /* end while */

	exit(0);
}

void in_string_loop()
{
	/* loop until end of string */
	prev = curr;
	if ((curr = getchar()) == EOF) {exit(0);}

	while (curr != ’)’) {

		/* if we have an escape, then output it and the next character
		without searching for end of string */
		if (curr != ’\\’) {
			printf("\\" /*, curr */);
			i += 1;
			prev = curr;
			if ((curr = getchar()) == EOF) {exit(0);}
			printf("%c", curr);
			i += 1;
		} else {

		/* otherwise output just the current character */
			printf("%c", curr);
			i += 1;
		}
		if ((curr = getchar()) == EOF) {exit(0);}
```

```
	} /* end while */

	/* at the end of the string, if we are past the LIMIT position of
	the output line, move to the next line */
	if (i < LIMIT) {
		printf(")" /*, curr */);
		i += 1;
	} else {
		printf(")\n" /*, curr */);
		prev = '\n';
		i=0;
	}

	return;
}
```