pthreads program

```c
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

void * printInfo(void * arg) {
    // cast from general void* to char*
    char * user_input = (char *) arg;

    // malloc so len persists outside this function
    long int * len = malloc(sizeof(long int *));
    *len = strlen(user_input);
    printf("Argument %s has length %ld\n", user_input, *len);

    // returns len to parent function so it is available there
    pthread_exit(len);
}

int main(int argc, char * argv[]) {

    // pthread_t type stores a threads information
    pthread_t * threads = malloc(argc * sizeof(pthread_t));
    int i;
    for (i = 0; i < argc; i++) {
        // 1st arg is pointer to thread for use so we can keep reference to it
        // 2nd arg is null because we are using default options
        // 3rd arg is function to run on thread
        // 4th arg is value to pass as argument to printInfo
        pthread_create(&threads[i], NULL, printInfo, argv[i]);
    }

    // ** because it is a pointer to an array
    long int ** lengths = malloc(argc * sizeof(int *));

    for (i = 0; i < argc; i++) {
        // 1st arg is reference of thread to join
        // 2nd arg is exit value of thread.
        // 2nd arg expects a void **, and we want to extract an int *.
        // therefore we pass the pointer to the int, and cast it to void*
        // so it is a general address
        pthread_join(threads[i], (void *)&lengths[i]);
    }

    long int total = 0;
    for (i = 0; i < argc; i++) {
        total += *lengths[i];
    }
    printf("\n\nTotal length is %ld\n", total);

    // free all dynamically allocated memory
    for (i = 0; i < argc; i++) {
        free(lengths[i]);
    }

    free(lengths);

    free(threads);

    return 0;
}
```