CS101 Lab-6

Today's Linux Commands

- Work in a directory named `~/lab6g1`
- We've learned how to redirect input from a command (`|`), a file (`<`) and how to redirect output to a file (`>`). We can also time our programs to see how long it takes to run.
- Today, we'll learn about file permissions on Linux. These determine who can read, write, or execute a file. Crucial for maintaining security and controlling access to your files and directories!
- Three main types of permissions:

  Read (r): Allows viewing the contents of a file or listing the contents of a directory.

  Write (w): Allows modifying the contents of a file or creating/deleting files within a directory.

  Execute (x): Allows executing a program (i.e. a file) or accessing a directory (to enter it).

- Permissions are assigned to three categories of users:

  Owner (u): The user who owns the file or directory.

  Group (g): Users who are members of the file's group.

  Others (o): All other users on the system.

- You can view file permissions using the `ls -l` command. The output will show a string of ten characters, like `-rwxr-xr--`. Output indicates:
  - The first character indicates the file type (e.g., `-` for a regular file, `d` for a directory).
  - The next nine characters are grouped into sets of three, representing the permissions for the owner, group, and others, respectively.
- You can change file permissions using the chmod command, either with symbolic modes (e.g., `chmod u+w filename`) or octal codes (i.e., viewing permissions as 3 bit octal number) (e.g., `chmod 755 filename`).
- Notice that all compiled executable files have `x` permission.

Instructions for the lab assignment:

Solve the **first three problems** in the lab. The other problems are for homework.

1. Ask the user for the array size. Given an array of integers of that size and a target value, write a program that finds two numbers in the array that add up to the target value and prints their indices. If no such numbers exist, report so.

2. Copy the above program and modify your code to find up to three numbers in the array that add up to the target value and print their indices. If no such numbers exist, report so.

3. Write a program to take as an input a 3x3 matrix M. Compute and print 2M, M^2, and M^T.

4. Ask the user for the array size. Given an array of integers of that size and an index, delete the element at that index. Shift the remaining elements towards the left accordingly.

5. **Optional Challenge:** Copy the program from Problem 1 and modify your code to take as input a parameter 1 <= k <= array_size and find k numbers in the array that add up to the target value and print their indices. If no such numbers exist, report so. You may assume that the array_size is at most 20. Hint: Bitmasks provide a programmer-friendly way to iterate over all possible subsets of a set. Learn about and use them!

Solution for Lab-5 Problem 2

Use proper indentations, comments, and descriptive naming of variables.

```c
#include <stdio.h>
int main() {
    long int N;
    printf("Enter a positive integer: ");
    scanf("%ld", &N);
    if(N <= 0) {
        printf("Invalid input. Please enter a number greater than 0.\n");
        return 0;
    } else if (N == 1) { //1 is a special case
        printf("1 is neither prime nor composite");
        return 0;
    }
    int composite_flag = 0; //will be set to 1 if N is found composite
    for(long int i=2; i*i <= N; i++) {
        if(N%i == 0) {
            composite_flag = 1;
            break;
        }
    }
    if(composite_flag) printf("%ld is composite.\n", N);
    else printf("%ld is prime.\n", N);
    return 0;
}
```