Project8 - Advanced Programming Workshop

# Overview

This project was developed as part of an advanced programming workshop assignment. It includes several functions that demonstrate key concepts like:  
- String manipulation  
- File handling  
- Dynamic memory allocation  
- Text decoding

# Features

## 1. Split Function

Description:  
The `split()` function receives a string and a letter. It splits the string into words and returns an array of words that start with the given letter. The memory for the array is allocated dynamically.

Function Signature:  
`char \*\*split(char letter, char \*str, int \*p\_size)`

Example:  
For `str = "Rony Goodman got a good mark"` and `letter = 'g'`, the function will return the following array:  
Goodman  
got  
good

Memory Management:  
The function allocates memory dynamically and returns the pointer to the array. If no words match the condition, the function returns a `NULL` pointer.

## 2. File Creation and Letter Analysis

Part A:  
The function `createFile(char \*filename)` creates a text file with the specified name and writes a sequence of characters into it.

Part B:  
The function `commonestLetter(char\* filename)` reads the content of the text file and finds the most common letter (case insensitive). If multiple letters have the same frequency, it returns the letter closest to the end of the alphabet.

Function Signatures:  
- `void createFile(char \*filename)`  
- `char commonestLetter(char\* filename)`

Example:  
For a file containing the text:  
d=AB+C-a/(b+c+a+bc)  
D=2-a-b  
c=6+d  
The function will return 'C', because it appears the most number of times, and its ASCII value is larger than the other letters.

## 3. Text Decoding

Description:  
The `decode()` function decrypts a given text using a specific algorithm. For each word in the text:  
- Subtract 1 from the ASCII value of the first character.  
- Subtract 2 from the ASCII value of the second character.  
- Continue this pattern for the remaining characters.

Function Signature:  
`void decode(char \*text)`

Example:  
Input text: `"d 8bc"`  
Decoded text: `"c a5a"`

## Additional Notes

- The project also includes memory management functions for dynamic allocation, such as:  
 - `freeMatrix()`: For freeing dynamically allocated memory.  
 - `printStringsArray()`: For displaying dynamic arrays.  
- The input/output handling and user-friendly messages guide the user through the program's operations.

# Compilation and Usage

To compile the program, make sure to include all necessary files and libraries. You can use the following command to compile:  
`gcc -o Project8 Assignment\_2\_template.c`  
After compiling, run the program and follow the on-screen instructions.

# Requirements

The project was developed using C, and it requires standard C libraries for:  
- File handling  
- Dynamic memory allocation  
- String manipulation  
Ensure sufficient memory for dynamic allocation, as this project relies on this extensively.

# License

This project is part of an academic course and is intended for educational purposes only.