Project 6, Program Design

1. (100 points) One of the most basic ways of concealing a message is to hide a message in a larger passage that makes sense. Write a program that prompts the user to enter the name of a file for the content to be decrypted:

```
Enter the file name: message.txt

Output file name: message.txt.dcf
```

The program reads the content of the file and extract the first letter of each word, then writes the resulting message to a file with the same name but an added extension of `.dcf`. In this example, the original file name is `message.txt`, so the decrypted message will be stored in a file named `message.txt.dcf`.

Assume the file name is no more than 100 characters. Assume each word in the file is no more than 100 characters. Assume there are no more than 1000 words in the file.

Your program should include the following function:

```
void extract(char words[][101], int num_words, char
*result);
```

The function expects `words` to be an array of strings containing the content in the input file, `num_words` to be the number of strings in the array, and `result` to point to a string containing the output as a string.

Before you submit:

1. Compile with –Wall. Be sure it compiles on ***student cluster*** with no errors and no warnings.

gcc –Wall file_extract.c

2. Be sure your Unix source file is read & write protected. Change Unix file permission on Unix:

chmod 600 file_extract.c

3. Test your program with the shell scripts on Unix:

chmod +x try_extract
./try_extract

Total points: 100

1. A program that does not compile will result in a zero.
2. Runtime error and compilation warning 5%
3. Commenting and style 15%
4. Functionality 80%

Programming Style Guidelines

The major purpose of programming style guidelines is to make programs easy to read and understand. Good programming style helps make it possible for a person knowledgeable in the application area to quickly read a program and understand how it works.

1. Your program should begin with a comment that briefly summarizes what it does. This comment should also include your **name**.
2. In most cases, a function should have a brief comment above its definition describing what it does. Other than that, comments should be written only *needed* in order for a reader to understand what is happening.
3. Information to include in the comment for a function: name of the function, purpose of the function, meaning of each parameter, description of return value (if any), description of side effects (if any, such as modifying external variables)
4. Variable names and function names should be sufficiently descriptive that a knowledgeable reader can easily understand what the variable means and what the function does. If this is not possible, comments should be added to make the meaning clear.
5. Use consistent indentation to emphasize block structure.
6. Full line comments inside function bodies should conform to the indentation of the code where they appear.
7. Macro definitions (#define) should be used for defining symbolic names for numeric constants. For example: `#define PI 3.141592`
8. Use names of moderate length for variables. Most names should be between 2 and 12 letters long.
9. Use underscores to make compound names easier to read: `tot_vol` or `total_volumn` is clearer than totalvolumn.