CS 31 Worksheet 4

This worksheet is entirely **optional**, and meant for extra practice. Some problems will be more challenging than others and are designed to have you apply your knowledge beyond the examples presented in lecture, discussion or projects. All exams will be done on paper, so it is in your best interest to practice these problems by hand and not rely on a compiler.

Concepts

C-strings, 1-D Arrays

Reading Problems

1)

a) What does the following program print out?

```
#include <iostream>
using namespace std;

int main () {
    char phrase[] = "How the turntables.";

    for (int i = 0; i < strlen(phrase); i++) { // change in part b
        phrase[strlen(phrase) - 1] = '\0';
    }

    cout << "Result: " << phrase << endl;
}
```

b) Repeat part a), but replace the `for` loop inside `main()` with the following code:

```
int n = strlen(phrase);
for (int i = 0; i < n; i++) { // change in part b
    phrase[strlen(phrase) - 1] = '\0';
}
```

2) What does the following program print out?

```
int main() {
    char names[34] = "Anthony Millie Sophia Nina Kyle ";
    int count = 0;
    for (int i = 0; names[i] != '\0'; i++)    {
        int j = i;

        if (count == 0){
            for (j = i; names[j] != '\0' && names[j] != ' '; j++){
                if (j % 2 == 0)
                    cout << names[j];
            }
        }
        else if (count == 2){
            for (j = i; names[j] != '\0' && names[j] != ' '; j++){
                if (j % 2 == 0)
                    cout << names[j-1];
            }
        }
        else if (count == 1){
            cout << names[j+1];
            for (j = i; names[j] != '\0' && names[j] != ' '; j++);
        }
        else {
            cout << names[j];
            for (j = i; names[j] != '\0' && names[j] != ' '; j++);
        }

        i = j;
        count++;
    }
    cout << endl;
}
```

Programming Problems

1) Write a function *charInsert* that inserts a character into a valid C-string at a given position. The function has the following header:

```
bool charInsert(char str[], int n, int ind, char c)
```

The parameter `n` denotes the size of the character array `str`, which is not necessarily equivalent to the string's length. The insertion cannot be performed if ind is negative or greater than the string's length. Additionally, the insertion cannot be performed if the result would exceed the length of the array.

If the insertion is successful, the function returns true. If the insertion cannot be done, the function returns false and leaves `str` unmodified.

Examples:
```
char success[10] = "aaaaa";
bool res = charInsert(success, 10, 1, 'b'); // res should equal true
cout << success << endl; // abaaaa

char success[10] = "aaaaa";
bool res = charInsert(success, 10, 5, 'b'); // res should equal true
cout << success << endl; // aaaaab

char failure[6] = "aaaaa";
bool res = charInsert(failure, 6, 1, 'b'); // res should equal false
cout << failure << endl; // aaaaa
```

2) Write a function cReverse that reverses a valid C-string. The function has the following header:

```
void cReverse(char str[])
```

Example:
```
char test[9] = "American";
cReverse(test); // test should now store "naciremA"
```

3) Write an implementation of *strcat*, which concatenates two C-strings and has the following function header:

```
void strcat(char str1[], char str2[])
```

Assume there is enough space to save the entire result into str1.

Example:
```
char str1[20] = "Hello";
char str2[8] = " World!";
strcat(str1, str2);
cout << str1; // Hello World!
```

4) Write a function with the following header:

```
void eraseDuplicates(char str[])
```

This function should erase all duplicated characters in the string, so that only the first copy of any character is preserved. Feel free to use helper functions.

Example:
```
char test[50] = "memesformeforfree123";
```

```
eraseDuplicates(test); // test should now store "mesfor123"
```

5) Write a function wordShiftLeft that takes in a valid C-string and shifts each word left one character. A word is defined as a substring separated by spaces. Each shifted word wraps around, meaning that “CS31” would become “S31C”. The function has the following header:

```
void wordShiftLeft(char str[])
```

Example:

```
char test[] = "I.love.CS31";
wordShiftLeft(test);
cout << test << endl; // ".love.CS31I"

char test[] = "I love CS31";
wordShiftLeft(test);
cout << test << endl; // "I ovel S31C"
```