Slip 23

Program 1: Bubble Sort and Insertion Sort using Fork

```
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>

void bubble_sort(int arr[], int n) {
    for (int i = 0; i < n-1; i++) {
        for (int j = 0; j < n-i-1; j++) {
            if (arr[j] > arr[j+1]) {
                int temp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = temp;
            }
        }
    }
}

void insertion_sort(int arr[], int n) {
    for (int i = 1; i < n; i++) {
        int key = arr[i];
        int j = i - 1;
        while (j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
            j = j - 1;
        }
        arr[j + 1] = key;
    }
}

void display(int arr[], int n) {
    for (int i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
}
```

```
int main() {
    int arr[] = {64, 34, 25, 12, 22, 11, 90};
    int n = sizeof(arr) / sizeof(arr[0]);
    pid_t pid = fork();

    if (pid == 0) {
        printf("Child Process: Bubble Sort\n");
        bubble_sort(arr, n);
        printf("Sorted Array: ");
        display(arr, n);
    }
    else if (pid > 0) {
        wait(NULL);  // Wait for child process to complete
        printf("Parent Process: Insertion Sort\n");
        insertion_sort(arr, n);
        printf("Sorted Array: ");
        display(arr, n);
    }
    else {
        printf("Fork failed!\n");
    }

    return 0;
}
```

Program 2 Shell with `count` Command

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

void count_lines(char *filename) {
    FILE *file = fopen(filename, "r");
    if (file == NULL) {
        printf("File %s not found.\n", filename);
        return;
    }

    int lines = 0;
    char ch;
```

```
    while ((ch = fgetc(file)) != EOF) {
        if (ch == '\n') lines++;
    }

    printf("Total lines: %d\n", lines);
    fclose(file);
}

void count_words(char *filename) {
    FILE *file = fopen(filename, "r");
    if (file == NULL) {
        printf("File %s not found.\n", filename);
        return;
    }

    int words = 0;
    char word[100];
    while (fscanf(file, "%s", word) != EOF) {
        words++;
    }

    printf("Total words: %d\n", words);
    fclose(file);
}

void count_chars(char *filename) {
    FILE *file = fopen(filename, "r");
    if (file == NULL) {
        printf("File %s not found.\n", filename);
        return;
    }

    int chars = 0;
    char ch;
    while ((ch = fgetc(file)) != EOF) {
        chars++;
    }

    printf("Total characters: %d\n", chars);
```

```
    fclose(file);
}

int main() {
    char command[100], *args[10];
    while (1) {
        printf("\nmyshell$ ");
        fgets(command, 100, stdin);
        command[strlen(command) - 1] = '\0';  // Remove newline

        char *token = strtok(command, " ");
        int i = 0;
        while (token != NULL) {
            args[i++] = token;
            token = strtok(NULL, " ");
        }
        args[i] = NULL;

        if (strcmp(args[0], "count") == 0) {
            if (strcmp(args[1], "l") == 0) {
                count_lines(args[2]);
            } else if (strcmp(args[1], "w") == 0) {
                count_words(args[2]);
            } else if (strcmp(args[1], "c") == 0) {
                count_chars(args[2]);
            }
        } else if (strcmp(args[0], "exit") == 0) {
            exit(0);
        } else {
            int pid = fork();
            if (pid == 0) {
                execvp(args[0], args);
                exit(0);
            } else {
                wait(NULL);
            }
        }
    }
    return 0;
```

```
}
```