CSC209H Worksheet: Fork and Wait

Consider the program below that runs without errors. In the table to the right, indicate how many times the name of each fruit is printed.

```
int main() {
    int i;
    printf("Mangoes\n");
    int r = fork();
    printf("Apples\n");
    if (r == 0) {
        printf("Oranges\n");
        int k = fork();
        if (k >= 0) {
            printf("Bananas\n");
        }
    } else if (r > 0) {
        printf("Peaches\n");
        for(i = 0; i < 3; i++) {
            if ((r = fork()) == 0) {
                printf("Pears\n");
                exit(0);
                printf("Nectarines\n");
            } else if (r > 0) {
                printf("Plums\n");
            }
        }
    }
    return 0;
}
```

Fruit Name	Times Printed
Mangoes	
Apples	
Oranges	
Bananas	
Peaches	
Pears	
Nectarines	
Plums	

Several orderings of the fruit names are possible valid output. Some of these orderings even have the unix prompt displaying before the final fruit name (or names). Explain why this happens.

Not all of the fruit names could appear after the prompt in a valid output. For example the word `Mangoes` will never appear after the prompt. List all the fruit names that could occur after the prompt.

For this question you will write a program that forks one child for each command line argument. The child computes the length of the command line argument and exits with that integer as the return value. The parent sums these return codes and reports the total length of all the command line arguments together. For example if your program is called `spread_the_work` and is called as `spread_the_work divide the load` it prints `The length of all the args is 13`. We have provided some parts of the code and you must work within this framework and complete the missing pieces. You do **not** need to write `include` statements.

```
int main(int argc, char **argv) {
    // Declare any variables you need
    [                                        ]

    // Write the code to loop over the command line arguments.
    // (Remember to skip the executable name.)

    for [(                                       )] {

        // call fork
        [          ] = fork();

        if [(                              )] { // System call error

            // Handle the error
            [                                        ]

        } else if [(                              )] { // Child process

            // Child does work here

            [                                        ]
        }
    }


    // Parent process.
    // On the next page, finish the code to sum up
    // the return values from the child processes.
```

```
    printf("The length of all the args is %d\n", sum);
    return 0;
}
```