2b11.c

```
#include <stdio.h>          // Standard input/output library
#include <sys/types.h>      // Definitions for data types used in system calls
#include <unistd.h>         // UNIX standard functions
#include <stdlib.h>         // Standard library for memory allocation and process control
#include <string.h>         // String handling functions

int main(int cnt, char *arr[]) {
    char temp[10];          // Temporary array for sorting strings
    int n;                  // Variable to hold user input

    // Bubble sort to sort the command-line arguments
    for (int i = 1; i < cnt - 1; i++) {
        for (int j = 1; j < cnt - i; j++) {
            if (strcmp(arr[j], arr[j + 1]) > 0) {
                strcpy(temp, arr[j]);
                strcpy(arr[j], arr[j + 1]);
                strcpy(arr[j + 1], temp);
            }
        }
    }

    // Print the sorted array of arguments
    printf("\nSorted Array:");
    for (int i = 1; i < cnt; i++) {
        printf(" %s", arr[i]);
    }
    printf("\n");

    // Create a child process
pid_t pid = fork();

if (pid < 0) {
    // Fork failed
    perror("Fork failed");
    return 1;
}
    // Child process: prompt for an integer input before execv
    printf("\nEnter integer to execute execv: ");
    scanf("%d", &n);

   // Execute `s.out` with sorted arguments
    execv("./s.out", arr);

    }
```

2b2.c

```
#include<stdio.h>          // Include standard input/output library
#include<sys/types.h>      // Include definitions for data types used in system calls
#include<unistd.h>         // Include UNIX standard functions

int main(int cnt, char *arr[]) // Main function takes argument count and array of arguments
{
        // Print the process ID of the current process
        printf("This is second process %d", getpid());

        // Indicate the beginning of the reversed array output
        printf("\nReversed array:\n");

        // Loop through the arguments in reverse order
        for(int i = cnt - 1; i >= 1; i--) // Start from the last argument and go to the first
        {
                // Print each argument in reverse order
                printf("\n%s", arr[i]);    // Print the current argument
        }
}
```