Assignment2B

```
// child_program.c
#include <stdio.h>

int main(int argc, char *argv[]) {
printf("Child received array in reverse order:\n");
for (int i = argc - 1; i > 0; i--) {
printf("%s ", argv[i]);
}
printf("\n");
return 0;
}

#include <stdio.h>
#include <stdlib.h>
#include <unistd.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 t = arr[j];
arr[j] = arr[j + 1];
arr[j + 1] = t;
}
}

int main() {
int n;
printf("Enter number of elements: ");
scanf("%d", &n);

int arr[n];
printf("Enter %d integers:\n", n);
for (int i = 0; i < n; i++) scanf("%d", &arr[i]);

bubble_sort(arr, n);

pid_t pid = fork();

if (pid < 0) {
perror("fork failed");
exit(1);
} else if (pid == 0) {
// Child process: exec child_program with array as args
char **args = malloc((n + 2) * sizeof(char *));
args[0] = "./child_pro";

for (int i = 0; i < n; i++) {
args[i + 1] = malloc(12);
```

```
snprintf(args[i + 1], 12, "%d", arr[i]);
}
args[n + 1] = NULL;

execve("./child_program", args, NULL);

perror("execve failed");
exit(1);
} else {
wait(NULL); // Parent waits for child
}

return 0;
}
```

==========

```
pict@mplab-11:~/Desktop/33168$ gcc -o child_program child_pro.c
pict@mplab-11:~/Desktop/33168$ gcc -o parent_program parent_pro.c
pict@mplab-11:~/Desktop/33168$ ./parent_program
Enter number of elements: 5
Enter 5 integers:
1
10
66
22
8
Child received array in reverse order:
66 22 10 8 1
```