# 22AIE202 – OPERATING SYSTEMS

# LABSHEET 5

Name : Anuvind MP

Roll no: AM.EN.U4AIE22010

---

1. Write a C Program that allows communication between parent and child process using ordinary pipes. The child should take an input (a String) from the user and supply it to the parent and the parent should change it to a string in uppercase and print it there.

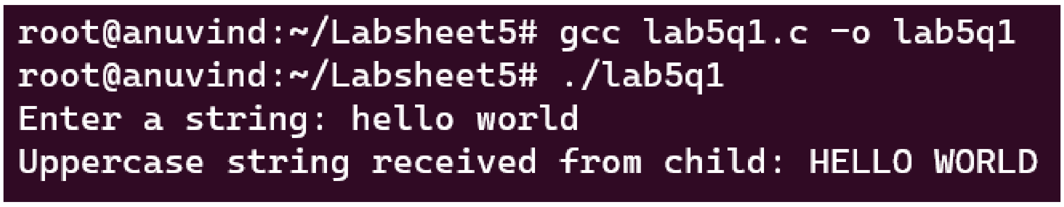
**CODE:**

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>
#include <ctype.h>

#define BUFFER_SIZE 1024

int main() {
    int pfd[2];
    char buffer[BUFFER_SIZE];
    pipe(pfd);
    pid_t pid = fork();
    if (pid == -1){
        perror("fork");
        exit(EXIT_FAILURE);}
    else if (pid == 0){
        close(pfd[0]); // Close unused read end
        printf("Enter a string: ");
        fgets(buffer, sizeof(buffer), stdin);
        write(pfd[1], buffer, sizeof(buffer));
        close(pfd[1]);} // Close write end
    else{
        close(pfd[1]); // Close unused write end
        read(pfd[0], buffer, sizeof(buffer));
        for (int i = 0; buffer[i]; i++){
            buffer[i] = toupper(buffer[i]);}
        printf("Uppercase string received from child: %s\n", buffer);
        close(pfd[0]); // Close read end
        wait(NULL);}
    return 0;}
```

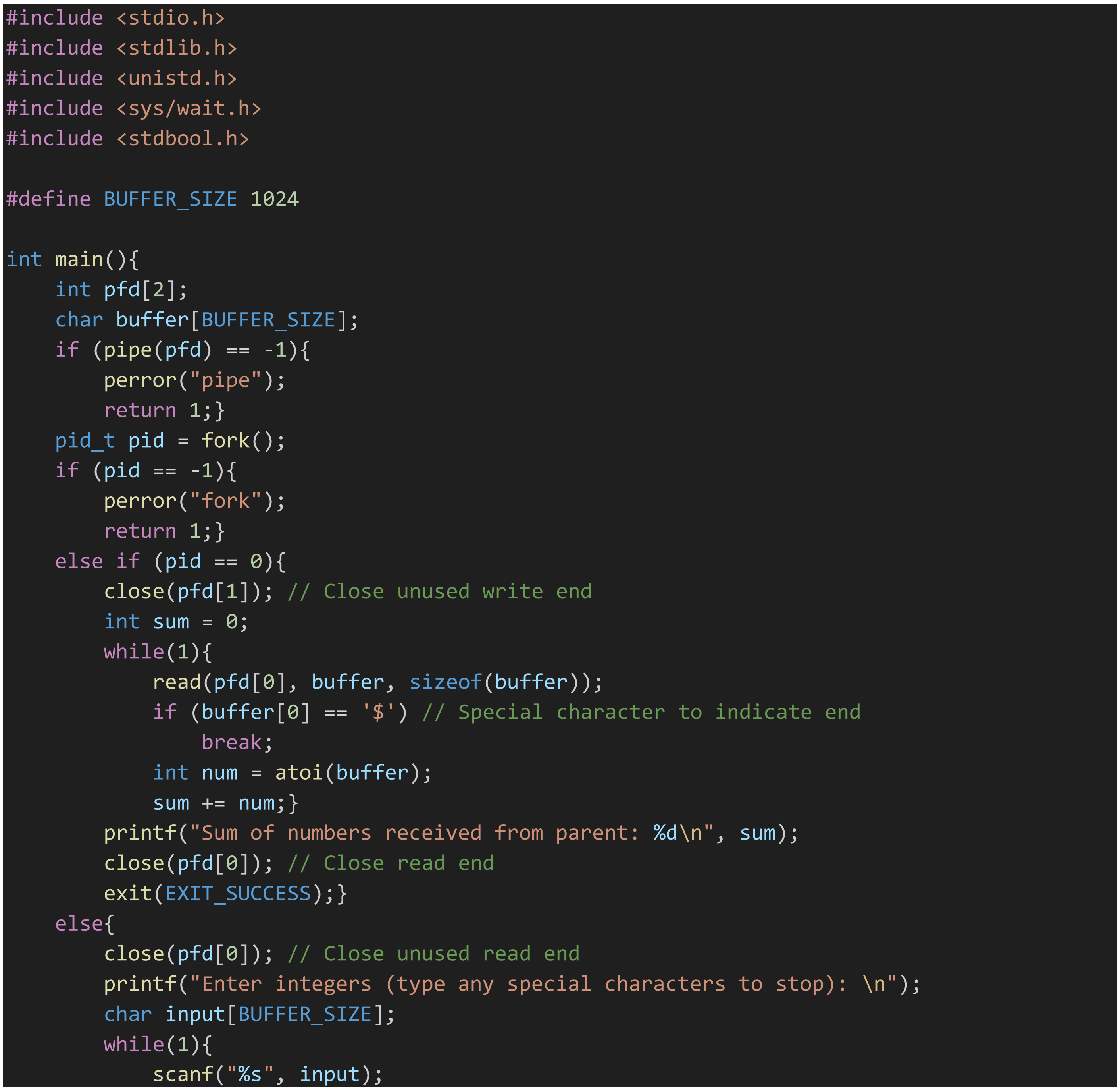
**OUTPUT:**



```
root@anuvind:~/Labsheet5# gcc lab5q1.c -o lab5q1
root@anuvind:~/Labsheet5# ./lab5q1
Enter a string: hello world
Uppercase string received from child: HELLO WORLD
```

2. Write a C Program that allows communication between parent and child process using ordinary PIPES. The parent should keep on taking integers from the user and supplying it to child until a special character is encountered. The child should display the sum of these numbers.

**CODE:**

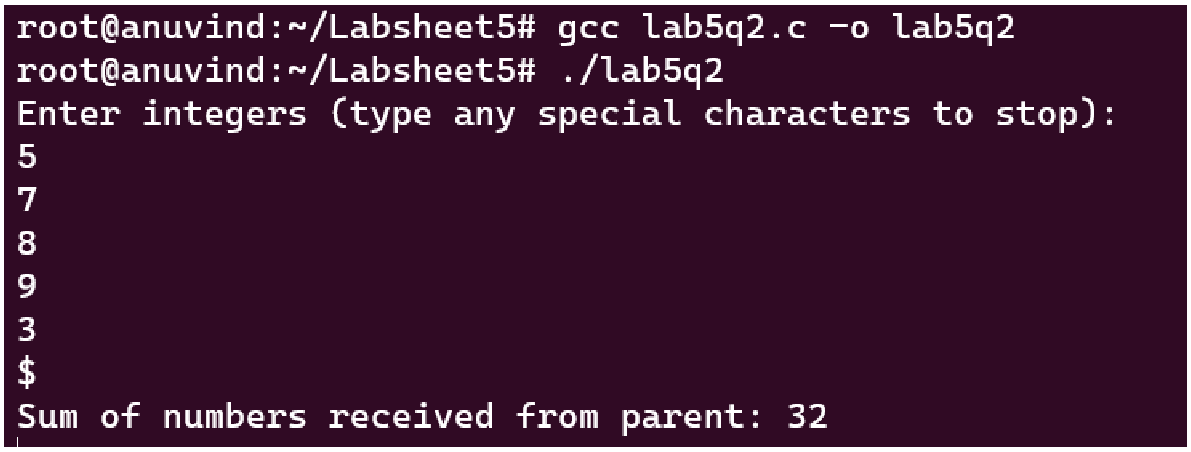


```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>
#include <stdbool.h>

#define BUFFER_SIZE 1024

int main(){
    int pfd[2];
    char buffer[BUFFER_SIZE];
    if (pipe(pfd) == -1){
        perror("pipe");
        return 1;}
    pid_t pid = fork();
    if (pid == -1){
        perror("fork");
        return 1;}
    else if (pid == 0){
        close(pfd[1]); // Close unused write end
        int sum = 0;
        while(1){
            read(pfd[0], buffer, sizeof(buffer));
            if (buffer[0] == '$') // Special character to indicate end
                break;
            int num = atoi(buffer);
            sum += num;}
        printf("Sum of numbers received from parent: %d\n", sum);
        close(pfd[0]); // Close read end
        exit(EXIT_SUCCESS);}
    else{
        close(pfd[0]); // Close unused read end
        printf("Enter integers (type any special characters to stop): \n");
        char input[BUFFER_SIZE];
        while(1){
            scanf("%s", input);
            write(pfd[1], input, sizeof(input));
            if (buffer[0] == '$') // Special character to indicate end
                break;}
        close(pfd[1]); // Close write end
        wait(NULL);} // Wait for child to finish
    return 0;}
```

**OUTPUT:**



```
root@anuvind:~/Labsheet5# gcc lab5q2.c -o lab5q2
root@anuvind:~/Labsheet5# ./lab5q2
Enter integers (type any special characters to stop):
5
7
8
9
3
$
Sum of numbers received from parent: 32
```

3. Write a c program using pipes to find average of square of numbers supplied by a user using 3 processes. 1 parent and two children.

a. Parent should continuously take integers as input from the user until a special character, square it and supply it to both children.

b. Child #1 should find sum of these numbers, send it to the parent and exit.

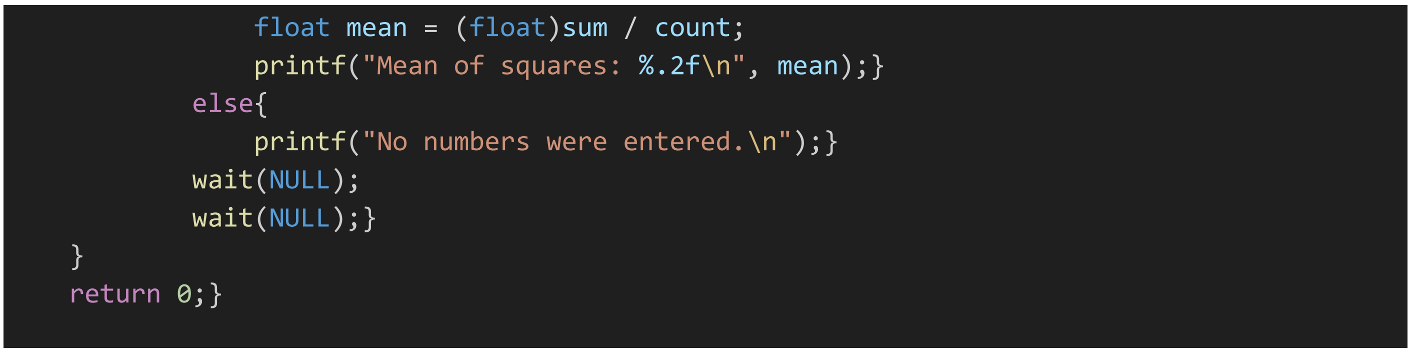
c. Child #2 should count these numbers, send them to the parent, and exit

d. Parent on getting response from both the children should find mean of square of numbers supplied by the user by dividing the child #1's result with child 2's and give it to the user

**CODE:**

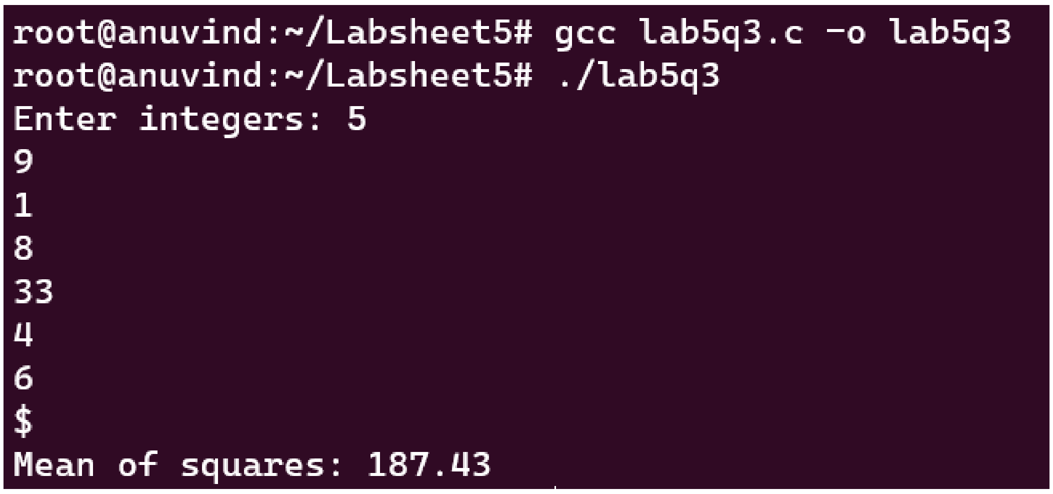
```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>

int main() {
    int pc1[2], pc2[2], cp1[2], cp2[2];
    if (pipe(pc1) == -1 || pipe(pc2) == -1 || pipe(cp1) == -1 || pipe(cp2) == -1){
        perror("pipe");
        return 1;}
    pid_t child1_pid, child2_pid;
    if ((child1_pid = fork()) == -1){
        perror("Fork failed");
        return 1;}
    if (child1_pid == 0){  // Child 1 process
        close(pc1[1]);
        close(cp1[0]);
        close(pc2[0]);
        close(pc2[1]);
        close(cp2[0]);
        close(cp2[1]);
        int sum = 0, num;
        while (read(pc1[0], &num, sizeof(int)) > 0){
            sum += num * num;}
        close(pc1[0]);
        write(cp1[1], &sum, sizeof(int));
        close(cp1[1]);
        exit(0);}
    else {  // Parent process
        if ((child2_pid = fork()) == -1){
            perror("Fork failed");
            return 1;}
        if (child2_pid == 0){  // Child 2 process
            close(pc1[0]);
            close(pc1[1]);
            close(cp1[0]);
            close(cp1[1]);
            close(pc2[1]);
            close(cp2[0]);
            int count = 0, num;
            while (read(pc2[0], &num, sizeof(int)) > 0){
                count++;}
            close(pc2[0]);
            write(cp2[1], &count, sizeof(int));
            close(cp2[1]);
            exit(0);
        }
        else{  // Parent process
            close(pc1[0]);
            close(pc2[0]);
            close(cp1[1]);
            close(cp2[1]);
            int num;
            printf("Enter integers: ");
            while (scanf("%d", &num) == 1) {
                write(pc1[1], &num, sizeof(int));
                write(pc2[1], &num, sizeof(int));}
            close(pc1[1]);
            close(pc2[1]);
            int sum, count;
            read(cp1[0], &sum, sizeof(int));
            read(cp2[0], &count, sizeof(int));
            close(cp1[0]);
            close(cp2[0]);
            if (count != 0){
```



```c
                float mean = (float)sum / count;
                printf("Mean of squares: %.2f\n", mean);}
            else{
                printf("No numbers were entered.\n");}
            wait(NULL);
            wait(NULL);}
    }
    return 0;}
```

**OUTPUT:**



```
root@anuvind:~/Labsheet5# gcc lab5q3.c -o lab5q3
root@anuvind:~/Labsheet5# ./lab5q3
Enter integers: 5
9
1
8
33
4
6
$
Mean of squares: 187.43
```