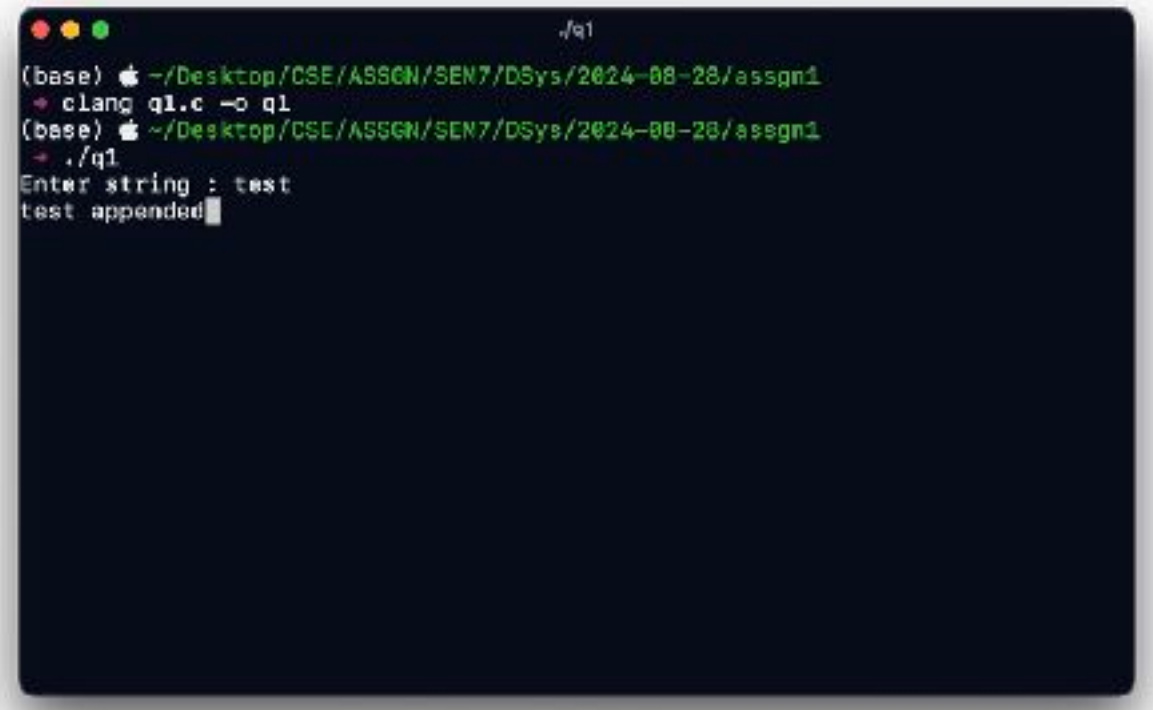
ASSIGNMENT 1
02/09/24

NAME : SHRESTH SONKAR
REGNO : 20214272
GROUP : CS7D
TOPIC : DISTRIBUTED SYSTEM
CODE : CS-17201

```c
//Shresth Sonkar
//20214272
//Q1 : Concat a string to entered string using parent
and child process

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/file.h>

#define MAX_SIZE 100

int main() {
    int fd[2];
    pid_t pid;
    char inputStr[MAX_SIZE];
    printf("Enter string : ");
    scanf("%s", inputStr);
    char appendStr[] = " appended";
    char buffer[MAX_SIZE];

    if (pipe(fd) == -1) {
        perror("Pipe failed");
        return 1;
    }

    pid = fork();

    if (pid < 0) {
        perror("Fork failed");
        return 1;
    }

    if (pid > 0) {
        close(fd[0]);
        write(fd[1], inputStr, strlen(inputStr) + 1);
        close(fd[1]);
        wait(NULL);

        fd[0] = open("/dev/fd/0", O_RDONLY);
        read(fd[0], buffer, sizeof(buffer));
        printf("Concatenated String: %s\n", buffer);
        close(fd[0]);
```

```c
    } else {
        close(fd[1]);
        read(fd[0], buffer, sizeof(buffer));

        int len = 0;
        while (buffer[len] != '\0') {
            len++;
        }

        int j = 0;
        while (appendStr[j] != '\0') {
            buffer[len] = appendStr[j];
            len++;
            j++;
        }
        buffer[len] = '\0';

        fd[1] = open("/dev/fd/1", O_WRONLY);
        write(fd[1], buffer, strlen(buffer) + 1);

        close(fd[0]);
        close(fd[1]);
        exit(0);
    }

    return 0;
}
```



```c
//Shresth Sonkar
//20214272
//Q2 : Enter matrix into parent, send to child and
calculate sum. Print result on parent using pipe

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

#define MAX_SIZE 100

void readMatrix(int rows, int cols, int matrix[rows]
[cols]) {
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            printf("[%d][%d]: ", i, j);
            scanf("%d", &matrix[i][j]);
        }
    }
}

void printMatrix(int rows, int cols, int matrix[rows]
[cols]) {
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            printf("%d ", matrix[i][j]);
        }
        printf("\n");
    }
}

int main() {
    int fd1[2], fd2[2];
    pid_t pid;

    if (pipe(fd1) == -1 || pipe(fd2) == -1) {
        perror("pipe failed");
        return 1;
    }

    pid = fork();

    if (pid < 0) {
```

```
        perror("fork failed");
        return 1;
    }

    if (pid > 0) {
        close(fd1[0]);
        close(fd2[1]);

        int rows, cols;
        printf("Enter the number of rows and columns of
the matrices: ");
        scanf("%d %d", &rows, &cols);
        int matrix1[rows][cols];
        int matrix2[rows][cols];

        printf("Enter elements of the first matrix:
\n");
        readMatrix(rows, cols, matrix1);
        printf("Enter elements of the second matrix:
\n");
        readMatrix(rows, cols, matrix2);


        write(fd1[1], &rows, sizeof(int));
        write(fd1[1], &cols, sizeof(int));
        write(fd1[1], matrix1, sizeof(int) * rows *
cols);
        write(fd1[1], matrix2, sizeof(int) * rows *
cols);

        int result[rows][cols];
        read(fd2[0], result, sizeof(int) * rows *
cols);
        printf("Sum of the matrices:\n");
        printMatrix(rows, cols, result);

        close(fd1[1]);
        close(fd2[0]);

        wait(NULL);

    } else {
        close(fd1[1]);
        close(fd2[0]);
```

```
        int rows, cols;
        read(fd1[0], &rows, sizeof(int));
        read(fd1[0], &cols, sizeof(int));

        int matrix1[rows][cols];
        int matrix2[rows][cols];
        read(fd1[0], matrix1, sizeof(int) * rows *
cols);
        read(fd1[0], matrix2, sizeof(int) * rows *
cols);

        int result[rows][cols];
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                result[i][j] = matrix1[i][j] +
matrix2[i][j];
            }
        }

        write(fd2[1], result, sizeof(int) * rows *
cols);
        close(fd1[0]);
        close(fd2[1]);
        exit(0);
    }

    return 0;
}
```