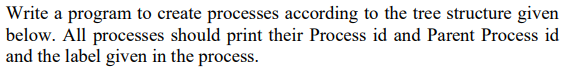
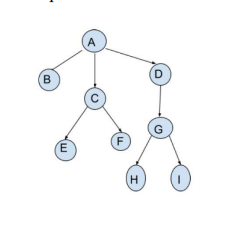
LAB SHEET 5

Name: Aadithyan Raju

Roll No: AM.EN.U4CSE21301

Q1.





CODE:

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/wait.h>

int main() {

    // Create process A

    pid\_t pid\_a = getpid();

    printf("Process A: pid=%d, ppid=%d, label=A\n", pid\_a, getppid());

    // Create process B

    pid\_t pid\_b = fork();

    if (pid\_b == 0) {

        printf("Process B: pid=%d, ppid=%d, label=B\n", getpid(), getppid());

        exit(0);

    }

    // Create process C

    pid\_t pid\_c = fork();

    if (pid\_c == 0) {

        printf("Process C: pid=%d, ppid=%d, label=C\n", getpid(), getppid());

        // Create process E

        pid\_t pid\_e = fork();

        if (pid\_e == 0) {

            printf("Process E: pid=%d, ppid=%d, label=E\n", getpid(), getppid());

            exit(0);

        }

        // Create process F

        pid\_t pid\_f = fork();

        if (pid\_f == 0) {

            printf("Process F: pid=%d, ppid=%d, label=F\n", getpid(), getppid());

            exit(0);

        }

        exit(0);

    }

    // Create process D

    pid\_t pid\_d = fork();

    if (pid\_d == 0) {

        printf("Process D: pid=%d, ppid=%d, label=D\n", getpid(), getppid());

        // Create process G

        pid\_t pid\_g = fork();

        if (pid\_g == 0) {

            printf("Process G: pid=%d, ppid=%d, label=G\n", getpid(), getppid());

            // Create process H

            pid\_t pid\_h = fork();

            if (pid\_h == 0) {

                printf("Process H: pid=%d, ppid=%d, label=H\n", getpid(), getppid());

                exit(0);

            }

            // Create process I

            pid\_t pid\_i = fork();

            if (pid\_i == 0) {

                printf("Process I: pid=%d, ppid=%d, label=I\n", getpid(), getppid());

                exit(0);

            }

            exit(0);

        }

        exit(0);

    }

    // Wait for all child processes to finish

    int status;

    wait(&status);

    wait(&status);

    wait(&status);

    wait(&status);

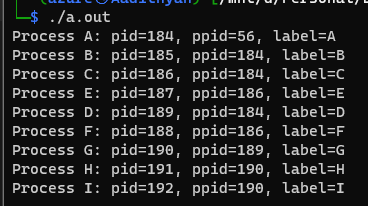
    wait(&status);

    wait(&status);

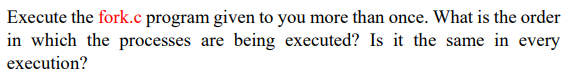
    return 0;

}

OUTPUT:



Q2.



Order: Child then Parent

Same: Yes it is the same.

Q3.



Code:

#include <stdio.h>

#include <sys/types.h>

#include <sys/wait.h>

#include <unistd.h>

#define MAX\_COUNT 5

void  ChildProcess(void);                /\* child process prototype  \*/

void  ParentProcess(void);               /\* parent process prototype \*/

void  main(void)

{

    pid\_t  pid;

    pid = vfork();

    if (pid == 0) ChildProcess();

    else ParentProcess();

    int status;

    wait(&status);

}

void  ChildProcess(void)

{

    int   i;

    for (i = 1; i <= MAX\_COUNT; i++) printf("   This line is from child, value = %d\n", i);

    printf("   \*\*\* Child process is done \*\*\*\n");

}

void  ParentProcess(void)

{

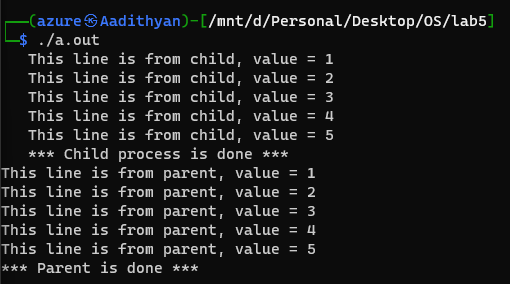
    int   i;

    for (i = 1; i <= MAX\_COUNT; i++) printf("This line is from parent, value = %d\n", i);

    printf("\*\*\* Parent is done \*\*\*\n");

}

OUTPUT:



Q4.



#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/wait.h>

#define PI 3.1415

void circle() {

    float r, a, p;

    printf("Enter r of the circle: ");

    scanf("%f", &r);

    a = PI \* r \* r;

    p = 2 \* PI \* r;

    printf("Area of the circle = %f\n", a);

    printf("Perimeter of the circle = %f\n", p);

}

void square() {

    float s, a, p;

    printf("Enter side length of the square: ");

    scanf("%f", &s);

    a = s \* s;

    p = 4 \* s;

    printf("Area of the square = %f\n", a);

    printf("Perimeter of the square = %f\n", p);

}

int main() {

    // Create process for circle

    pid\_t pid\_c = fork();

    if (pid\_c == 0) {

        circle();

        exit(0);

    } // Wait

    int status;

    waitpid(pid\_c, &status, 0);

    // Create process for square

    pid\_t pid\_s = fork();

    if (pid\_s == 0) {

        square();

        exit(0);

    }

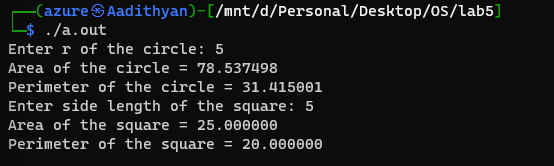
    // Wait

    waitpid(pid\_s, &status, 0);

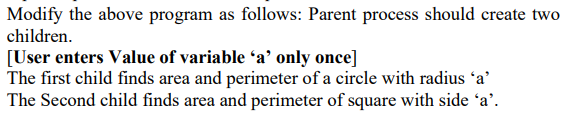
    return 0;

}

OUTPUT:



Q5:



CODE:

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/wait.h>

#define PI 3.1415

void circle(float r) {

    float a, p;

    a = PI \* r \* r;

    p = 2 \* PI \* r;

    printf("Area of the circle = %.2f\n", a);

    printf("Perimeter of the circle = %.2f\n", p);

}

void square(float s) {

    float a, p;

    a = s \* s;

    p = 4 \* s;

    printf("Area of the square = %.2f\n", a);

    printf("Perimeter of the square = %.2f\n", p);

}

int main() {

    // Create process for circle

    float a;

    printf("Enter a: ");

    scanf("%f", &a);

    pid\_t pid\_c = fork();

    if (pid\_c == 0) {

        circle(a);

        exit(0);

    }

    // Create process for square

    pid\_t pid\_s = fork();

    if (pid\_s == 0) {

        square(a);

        exit(0);

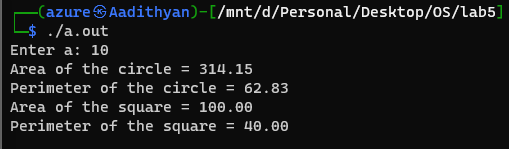
    }

    sleep(1);

    return 0;

}

OUTPUT



Q6.



#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/wait.h>

#define PI 3.1415

void circle(float r) {

    float a, p;

    a = PI \* r \* r;

    p = 2 \* PI \* r;

    printf("Area of the circle = %.2f\n", a);

    printf("Perimeter of the circle = %.2f\n", p);

}

void square(float s) {

    float a, p;

    a = s \* s;

    p = 4 \* s;

    printf("Area of the square = %.2f\n", a);

    printf("Perimeter of the square = %.2f\n", p);

}

int main() {

    // Create process for circle

    float a;

    printf("Enter a: ");

    scanf("%f", &a);

    pid\_t pid\_c = fork();

    if (pid\_c == 0) {

        circle(a);

        exit(0);

    } // Wait

    int status;

    waitpid(pid\_c, &status, 0);

    // Create process for square

    pid\_t pid\_s = fork();

    if (pid\_s == 0) {

        square(a);

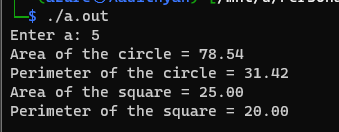
        exit(0);

    }

    return 0;

}

OUTPUT:



Q7.

CODE:

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/wait.h>

#define PI 3.1415

void circle(float r) {

    float a, p;

    a = PI \* r \* r;

    p = 2 \* PI \* r;

    printf("Area of the circle = %.2f\n", a);

    printf("Perimeter of the circle = %.2f\n", p);

}

void square(float s) {

    float a, p;

    a = s \* s;

    p = 4 \* s;

    printf("Area of the square = %.2f\n", a);

    printf("Perimeter of the square = %.2f\n", p);

}

int main() {

    // Create process for circle

    float a;

    printf("Enter a: ");

    scanf("%f", &a);

    pid\_t pid\_c = fork();

    if (pid\_c == 0) {

        circle(a);

        exit(0);

    }

    int status;

    waitpid(pid\_c, &status, 0);

    // Create process for square

    pid\_t pid\_s = fork();

    if (pid\_s == 0) {

        square(a);

        exit(0);

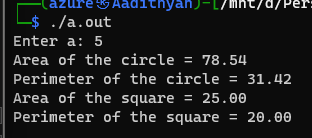
    }

    waitpid(pid\_s, &status, 0);

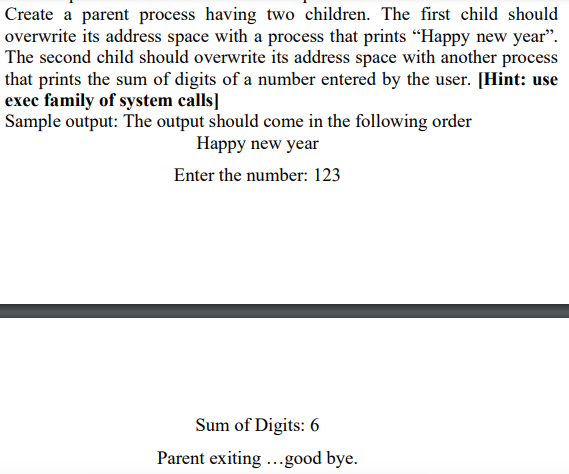
    return 0;

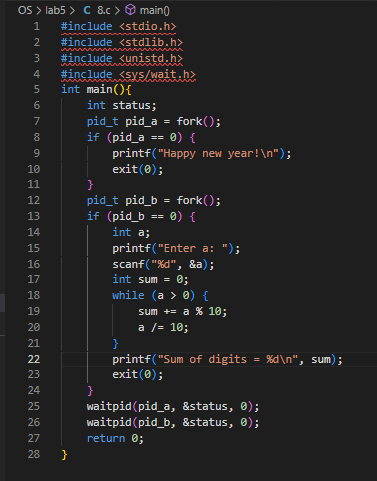
}

OUTPUT:



Q8.





OUTPUT:

