Names:

Rachel Peters - 816003509

Shivana Sookdeo - 816003030

Teneil Rodriguez - 816003952

Akil Hosang - 816003000

David Orr – 816003099

Course Code & Name: COMP 2604 – Operating Systems

Assignment 1

Year 2

Semester 2

**Question 1**

#include <stdio.h>

#include <sys/types.h>

#include <unistd.h>

#include <sys/wait.h>

int Collatz(int n){ /// This function allows for the Collatz Sequence of a number to be generated by taking in any number n

printf("%d ",n); /// the number is then printed

if(n==1) /// if n is one then it is returned

return 1;

else{

    if(n % 2 == 0) //else if the remainder of n divided by 2 is 0, then the function is called again with n divided by 2

        return Collatz(n/2);

    else /// otherwise it is called with n multiplied by 3 plus 1

        return Collatz((n\*3)+1);

}

}

int main(void){

pid\_t pid; /// a process id variable is declared

int status,n; /// an integer for status and n are declared

printf("Enter the number you wish to have the collatz sequence done on");

scanf("%d",&n); /// The number you wish to find the collatz sequence for is taken

pid = fork(); /// A child process is created and the pid is assigned a process id number

if(pid == 0){ /// if the pid is 0, that means it is a child process

if(n<1 || n>9){ /// if the n inputted is less than 1 or more than 9 then the code cannot be executed

printf("Cannot be executed");

}else{ /// otherwise it will call upon the collatz function with n

Collatz(n);

}

}else if(pid > 0){ /// if the pid is more than 0 then it is the parent process

   pid = wait(&status); /// the parent will then wait until the child process finishes executing and state that it is done executing when the child is finished

     printf("\nSequence Completed");

}

}

**Question 2**

1. FCFS

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0 15 18 27 33 37 | | | | | |
| P2 | P5 | P4 | P3 | P1 |  |

Average Waiting time = (15-0 + 18-0 + 27-0 + 33-0 + 37 -0)/ 5= 26

1. Non-pre-emptive Shortest Job First

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0 3 7 13 22 37 | | | | | |
| P2 | P5 | P4 | P3 | P1 |  |

Average Waiting time = (3-0 + 7-0 + 13-0 + 22-0 + 37 -0)/ 5= 16.4

1. Non-pre-emptive priority Scheduling

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0 15 21 25 28 37 | | | | | |
| P1 | P4 | P5 | P2 | P3 |  |

Average Waiting time = (15-0 + 21-0 + 25-0 + 28-0 + 37 -0)/ 5= 25.2

1. Round robin with a 5ms time quantum

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 5 8 13 18 22 27 31 32 37 | | | | | | | | | |
| P1 | P2 | P3 | P4 | P5 | P1 | P3 | P4 | P1 |  |

Average Waiting time = (37-0 + 8-0 + 31-0 + 32-0 + 22 -0)/ 5= 26

**Question 3**

There are child processes created when the program is run.

The output is the values 9 to 1 being printed 512 times

**Question 4**

There are 3 copies of the variable x which are:

X = 90, X = 80 and X = 70