Name : Pavankumar Padole
Roll No : 200240520063

Questions :

1. Parent and child process
   Write a C/CPP program to create a child process. Child should print its pid and its parent's pid and should exit by printing message as "Child Exiting …". Parent should print its pid and should exit by printing message as "Parent Exiting ..".
   (Hint(functions to be used) : getpid, fork, getppid)

Ans :-

```
 #include<unistd.h>
#include<stdlib.h>
#include<iostream>
#include<sys/types.h>
using namespace std;

int main()
{

       pid_t pid;
        pid=fork();

        if(pid==0)
        {
                //child process

                cout<<"pid of child process----"<<getpid();
                cout<<endl;
                cout<<"ppid of child process----"<<getppid();
                cout<<endl;
        }
        else if(pid>0)
        {
                //parent process

                cout<<"pid of parent process----"<<getpid();
                cout<<endl;
                cout<<"ppid of parent process----"<<getppid();
                cout<<endl;
        }
        else
        {
                cout<<"fork failed";
        }
        return 0;
}
```

2. Scheduling functions

   Write a program in C/CPP to check the scheduling policy used by the process and its priority.

   (Hint(functions to be used) : sched_getscheduler, getpid)

Ans:-

```
#include<iostream>
#include<unistd.h>
#include<stdlib.h>
#include<sched.h>

using namespace std;

int main()
{
    int a;
    a=sched_getscheduler(getpid());
    switch(a)
    {
        case SCHED_OTHER:cout<<"another scheduling poly"<<endl;
        break;
        case SCHED_RR:cout<<"round robin scheduler"<<endl;
        break;
        case SCHED_FIFO:cout<<"first in first out"<<endl;
        break;

    }
return 0;
}
```

3. Scheduling functions

   Write a program in C/CPP to get the current scheduling policy of the process. The program should change the scheduling policy to the other than current one. Program should report errors if it fails to set the new scheduling policy.

   (Hint(functions to be used) : sched_setscheduler, getpid)

Ans :-

```
#include<iostream>
#include<unistd.h>
#include<sched.h>
using namespace std;
int main()
{
    int a;
    a=sched_setscheduler(getpid(),SCHED_FIFO,0);
    if(a==0)
    {
        cout<<"priority set";
    }
    else
    {
        cout<<"priority not set";
    }
return 0;
}
```

4. Scheduling algorithm

    Write a program in C/CPP to take process name, its arrival time and execution/burst time as input.

    Use FCFS(non-preemptive) algorithm to calculate wait time of each process, average wait time, turnaround time of each process and average turnaround time.

    Calculation of time will start from the arrival time of first process.

    execution/burst time - Time required for execution of process

    Wait time of process = response time of process - arrival time process

    Response time of process : time at which process is scheduled to run

    Average wait time = (sum of wait time of each process) / (number of processes)

    Turnaround time of process = (finish/completion time of process) - (arrival time of process)

    Average turnaround time = (sum of turnaround time of each process) / (number of processes)

Sample Input

Process	arrival time	execution/burst time
P1	0	3
P2	2	5
P3	5	6

Sample Output

Process	Response time	Completion/finish time	Waiting time	Avg waiting time	Turnaround time	Avg turnaround time
P1	0	3	0-0 = 0	(0+1+3)/3=4/3	3-0=3	(3+6+9)/3=18/3=6
P2	3	8	3-2=1		8-2=6	
P3	8	14	8-5=3		14-5=9	

Ans :-

```
#include<iostream>
#include<cstring>
using namespace std;

class process{

    public:
        string name;
        int arrTime;
        int burstTime;
        int respTime;
        int compTime;
        int waitTime;
        int turnTime;
};

int main()
{
    int i,noproc,avgwaitTime,avgturnTime,sum1 = 0,sum2 = 0;
    cout<<"no of process=\n";
    cin>>noproc;
    process p[ noproc ];
    for(i=0;i<noproc;i++)
    {
        cout<<"\n enter the process "<<i<<" name = ";
        cin>>p[i].name;

        cout<<"\n enter the process "<<i<<" arrival time = ";
        cin>>p[i].arrTime;

        cout<<"\n enter the process "<<i<<" burst time = ";
        cin>>p[i].burstTime;

    }

    for(i=0; i<noproc; i++)
    {
if( i== 0){
            p[i].respTime = p[i].arrTime;
            p[i].compTime = p[i].respTime + p[i].burstTime;
            p[i].waitTime = p[i].respTime - p[i].arrTime;
            p[i].turnTime = p[i].compTime - p[i].arrTime;
        }
        else{

            p[i].respTime = p[i-1].respTime + p[i-1].burstTime;
            p[i].compTime = p[i].respTime + p[i].burstTime;
            p[i].waitTime = p[i].respTime - p[i].arrTime;
            p[i].turnTime = p[i].compTime - p[i].arrTime;
        }
    }


    for(i=0;i<noproc;i++)
    {
        sum1= sum1 + p[i].waitTime;
        sum2= sum2 + p[i].turnTime;
    }

    avgwaitTime = sum1/noproc;
    avgturnTime = sum2/noproc;

    for(i=0;i<noproc;i++)
```

```
        {
 cout<<"\n process no "<<i+1;
                cout<<"\n name "<<p[i].name;
                cout<<"\n arrival time "<<p[i].arrTime;
                cout<<"\n burst time "<<p[i].burstTime;
                cout<<"\n response time "<<p[i].respTime;
                cout<<"\n complete time "<<p[i].compTime;
                cout<<"\n waiting time "<<p[i].waitTime;
                cout<<"\n turnaround time "<<p[i].turnTime;
                cout<<endl;
        }
 cout<<"\n average waiting time = "<<avgwaitTime;
        cout<<"\n average turnaround time ="<<avgturnTime;
return 0;
}
```