Name: Amber khurshid
Section: BAI-5A
Roll no: 22P-9295

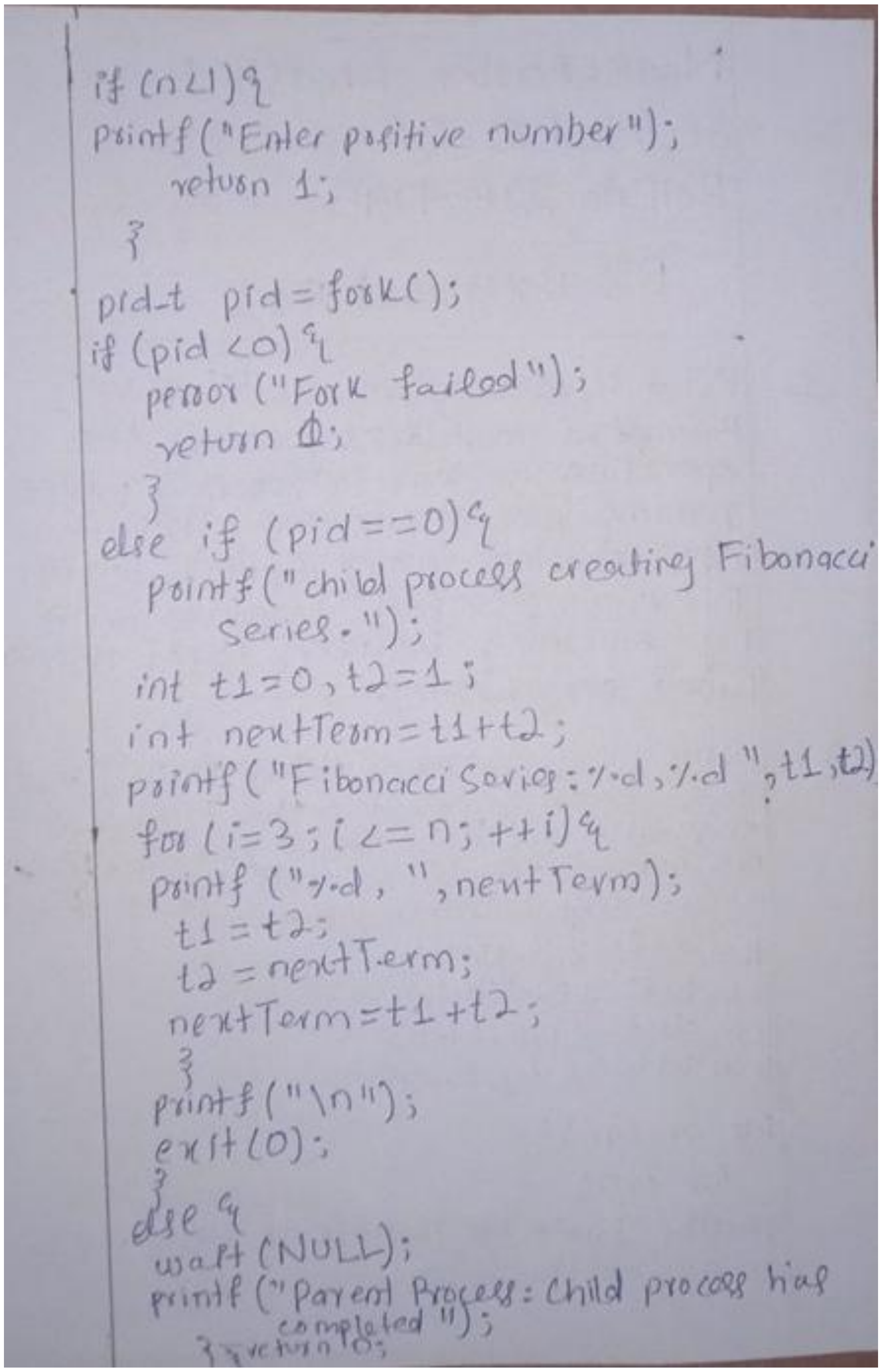
OS Assignment #02

1. What is preemptive multitasking?
Preemptive multitasking enables the operating system to forcibly pause a running process, ensuring efficient CPU sharing among multiple tasks. This ensures a fast response time by switching between tasks frequently based on priority.

2. Write a C program using fork() system call that generates the Fibonacci sequence in the child process. The number of sequence will be provided as an input from user.

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

int main(){
   int i,n;
printf("Enter the number of terms: ");
scanf("%d" &n);
```



```
if (n<1){
printf("Enter pofitive number");
    retusn 1;
 }
pid-t  pid = fork();
if (pid <0) {
   persor ("Fork failed");
   return 1;
  }
else if (pid==0){
  printf("child process creating Fibonacci
        series.");
  int t1=0, t2=1;
  int nextTerm=t1+t2;
  printf("Fibonacci Series: %d, %d ", t1, t2)
  for (i=3; i<=n; ++i) {
   printf ("%d, ", nextTerm);
    t1 = t2;
    t2 = nextTerm;
   nextTerm=t1+t2;
    }
  printf("\n");
  exit(0);
 }
else {
  wait (NULL);
  printf ("Parent Process: Child process has
            completed");
   }} return 0;
```

3. What is the main advantage of microkernel approach to system design? How do user programs and system services interact in a microkernel architecture? What are the disadvantages of using microkernal approach?

Main Advantage:

The main advantage of the microkernel design is its modularity, allowing easier system extension and more secure, reliable operation. System services communicate through message passing, but performance can suffer due to overhead in communication.

How user programs & system services interact:

An a microkernel architecture they interact via message passing, where user-space services communicate through the kernel to access hardware or perform system functions.

Disadvantages:

The disadvantages include performance overhead due to frequent context switching and message passing, which can reduce efficiency compared to monolithic kernels.

4. How could a system be designed to allow a choice of operating systems from which to boot?
What would the bootstrap program need to do?

1. Install multiple operating systems: Each OS is intalled on different partitions.
2. Configure a bootloader: A bootloader like GRUB or LILO can be installed, which recognizes the different operating systems.
3. Boot Menu: The bootloader displays a menu at startup listing the available OS options.
4. OS selection: The user selects an OS and the bootloader loads the appropriate kernel into memory and transfers control to it.

The bootstrap program must load the bootloader into memory, which in turn loads the chosen OS kernel and initializes the system.

5. When a process creates new process using fork() operation, which of the following are shared between the parent and child process?

a) Heap

Not shared. The child gets a copy of the parents heap, but they are independent of each other.

b) Stack

Not shared. The child has its own copy of stack.

c) shared memory segment

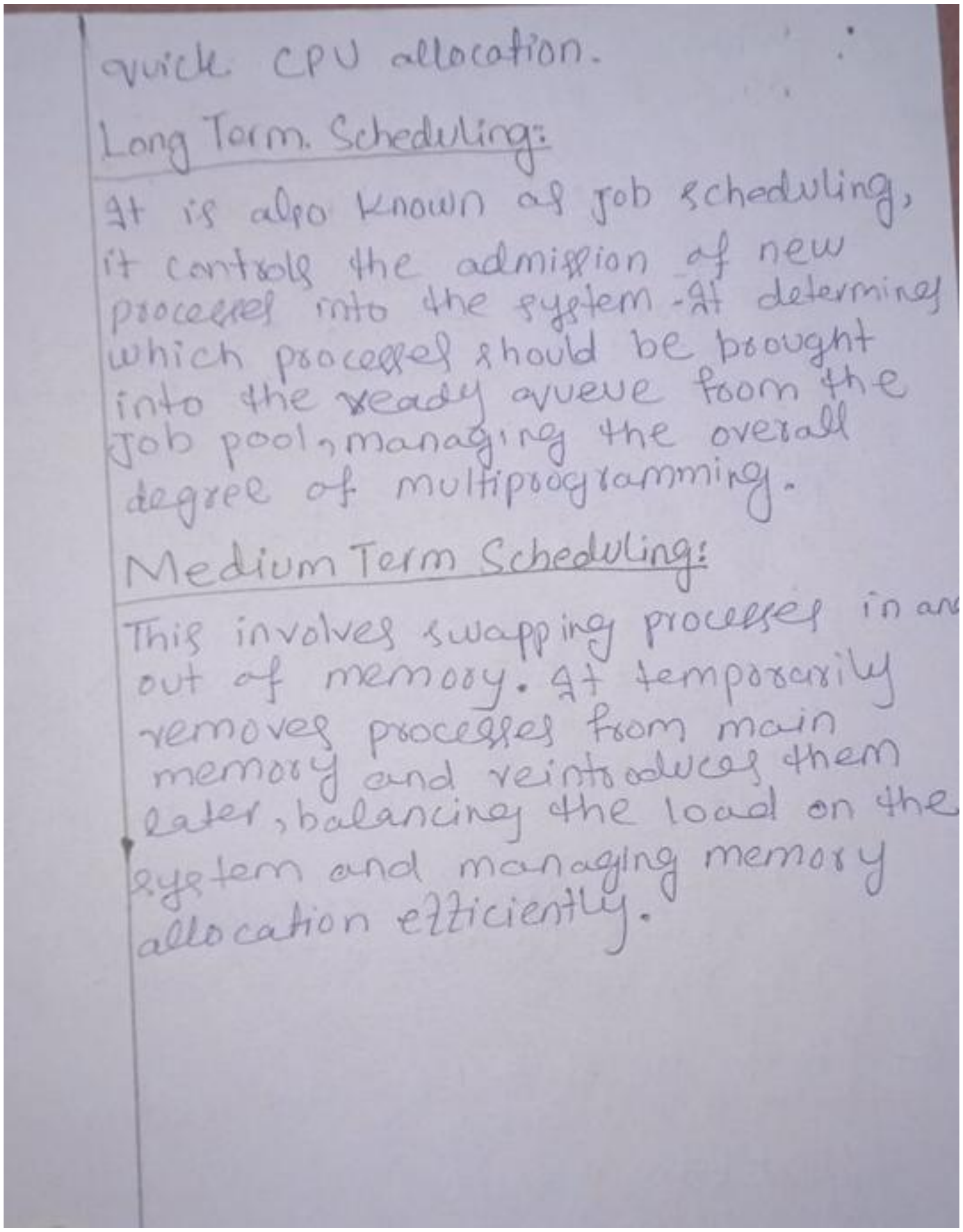
Shared. If there are shared memory segments, both parents and child can ~~child~~ access the same memory region.

Thus, only shared memory segments are shared between the parent and child processes.

6. What are short, long and medium-term scheduling?

Short term Scheduling:

It is also known as CPU schedulin, it decides which process in the ready queue will be executed nex by the CPU. It run frequently and focuses on process prioritization for


quick CPU allocation.

Long Term Scheduling:

It is also known as job scheduling, it controls the admission of new processes into the system - It determines which processes should be brought into the ready queue from the job pool, managing the overall degree of multiprogramming.

Medium Term Scheduling:

This involves swapping processes in an out of memory. It temporarily removes processes from main memory and reintroduces them later, balancing the load on the system and managing memory allocation efficiently.