National University of Computer and Emerging Sciences

FAST School of Computing **Fall-2023** **Islamabad Campus**

CS-2006: Operating Systems

Serial No:

Final Exam
Part-II Solution
Total Time: 2 Hours
Total Marks: 115

Thursday, 4th January, 2024

Course Instructors

Maryam Shahbaz, Muhammad Aadil ur Rehman

_____	_____	_____	_____
Student Name Student Signature	Roll No.	Course Section	Signature of Invigilator

DO NOT OPEN THE QUESTION BOOK OR START UNTIL INSTRUCTED.

Instructions:

1. Attempt on question paper. Attempt all of them. Read the question carefully, understand the question, and then attempt it.
2. No additional sheet will be provided for rough work. Use the back of the last page for rough work.
3. If you need more space write on the back side of the paper and clearly mark question and part number etc.
4. After asked to commence the exam, please verify that you have **Fourteen (14)** different printed pages including this title page. There are a total of **7** questions.
5. Calculator sharing is strictly prohibited.
6. Use permanent ink pens only. Any part done using soft pencil will not be marked and cannot be claimed for rechecking.

	Q-1	Q-2	Q-3	Q-4	Q-5	Q-6	Q-7	Total
Marks Obtained								
Total Marks	**15**	**25**	**10**	**10**	**20**	**20**	**15**	**115**

Question 1 [15 Marks]

Consider the C program given below and answer the following questions.

Note:

- Showing the rough work (dry run/process tree) in the space provided on the next page is mandatory, otherwise, no marks will be given.
- fflush() used in the program ensures immediate output flushing, avoiding buffering issues i.e., duplicate values due to std stream copies generated due to forks.
- There are no errors in the program.

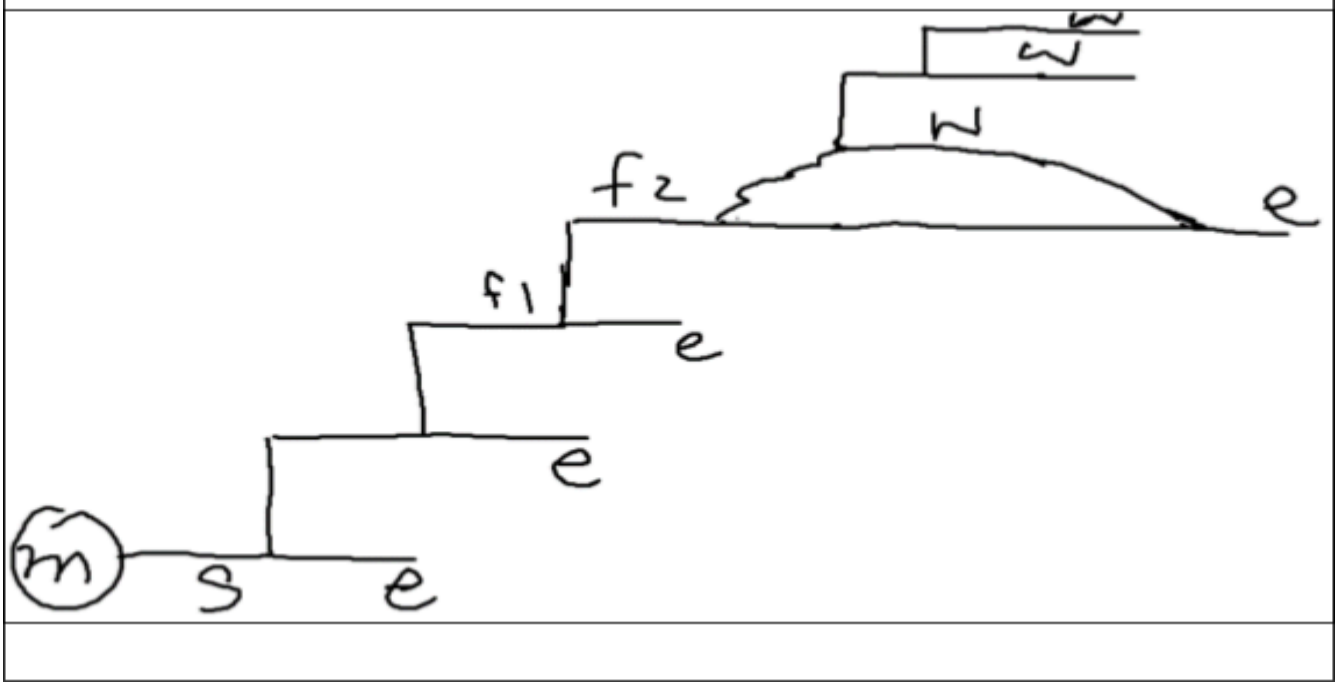
```
#include<stdio.h>
#include<unistd.h>
#include<pthread.h>

void *tFun(void *args){
      fork() && fork();
      printf("\n work");
      pthread_exit(NULL);
}

int main(int c, char* args[]) {
    pthread_t t;

    printf("\n start");
    fflush(stdout);

    if( !fork() && !fork()) {
        printf("\n f1 " );
        fflush(stdout);

        if( !fork() ) {
           printf("\n f2" );
           fflush(stdout);

           pthread_create(&t, NULL, tFun, NULL);
           pthread_join(thread, NULL);
        }
 }
 printf("\n end");
}
```

a). Using threads and forks simultaneously are prone to unexpected results if not carefully programmed, mention two reasons in bullet points? [2 marks] {**1** marks for each point, any of the following points can be choosen}

1. Threads and forks may share the same memory space. **Unintended modifications** by one can affect the data seen by the other.

2. Threads within a process share resources, and forks create separate processes. Precise coordination and synchronization mechanisms are needed to avoid **conflicts, race conditions , data corruptio, dead locks**.
3. Threads and forks may introduce overhead due to synchronization and context switching.
4. **Debugging** becomes challenging with concurrent threads and processes.Mixing threads and forks can result in **complex control flow**.
5. Incorrectly managed memory can lead to **memory leaks**.
6. It's possible for threads to create multiple child processes that become zombies. When using threads alongside forked processes

[Rough Work for Question 1]



b). Write the answer to the following questions if the program given above is executed. **[6 marks]**

Sr. #	Question	Answer
1	Total number of processes created including the main process that runs the program.	**6 (0.5 marks for 7 or 8)**
2	How many times "**start**" will be printed?	**1**
3	How many times "**end**" will be printed?	**4**
4	How many times "**f1**" will be printed?	**1**
5	How many times "**f2**" will be printed?	**1**
6	How many times "**work**" will be printed?	**3**

{**1** marks for each value}

c). Highlight(encircle) the logical issues in the following code snippet and explain each issue briefly. Consider there are no syntax issues, and all the libraries are included. [5 marks]

Code Snippet	Issues

`int count=0;`
`pthread_t tid[n];`
 	
`void* fun(void * val) {`
`count = count + val;`
`exit();`
`}`
 	
`int main(){`
`pthread_detach(pthread_self());`
 	
`int N = 5;`
`pthread_t tid[N];`
 	
`for(int i=0; i<N; i++) {`
`pthread_create($tid[i], NULL, fun, (void *) i);`
`}`
 	
`printf("Count = %d",count);`
 	
`pthread_exit(NULL);`
`}`
	1. Count is shared
	2. Iterator Is being passed
	3. Threads Not joined before printing count
	4. Main is detached
	5. Exit is being called in worker function
	6. $ instead of &
	7. global tid[n] is being initialized without specific value of n
	
	{1 marks for any of the above five issues}

d). How many processes will be create if the following code is executed, Assuming there are no errors and all the libraries are included? [2]

Code Snippet	Answer

`int main(){`
`fork();`
`if(fork()){`
`if(!fork()){`
`char *argv[2] = {"AnotherProgram", NULL};`
`execv (argv[0], argv);`
`}`
`}`
`}`
	6
	
	{**2 or ZERO** marks }

Question 2 [25 Marks]

Five processes A, B, C, D and E arrive in this order at the given time with the following CPU bursts. The process is immediately eligible for scheduling i.e. if a process arrives at 1 ms, it can be scheduled at 1 ms.

Process	CPU Burst (ms)	Arrival time (ms)	Priority (1 is lowest)
A	4	6	1
B	7	3	3
C	3	2	2
D	3	3	2
E	5	4	1
F	9	0	3

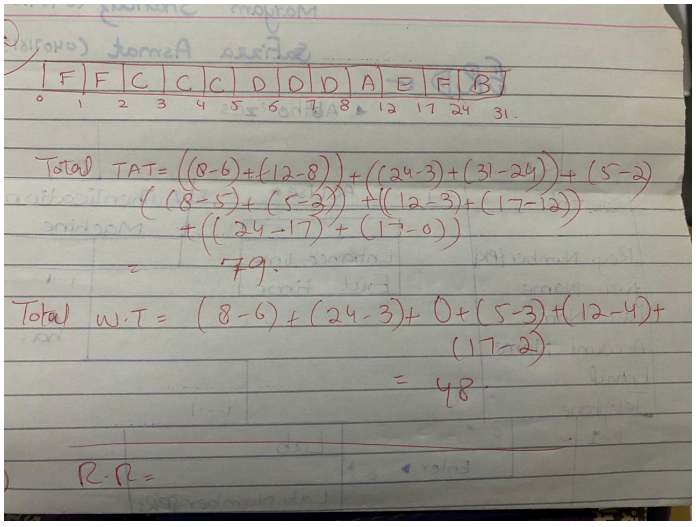
a). Fill in the entries of the following table with **total** turnaround time and **total** waiting time for each indicated scheduling policy *(**Note**: You do not have to calculate the **average** turnaround / waiting time).* Ignore context switching overhead. **All steps depicting the calculations are mandatory. Filling the table (even with correct entries) without correct solving will lead to zero marks in the question. Only correct entries in the table after proper calculations will secure your marks. No marks will be given for the direct answers. [18 marks]**

Algorithm	**Total** turnaround time (ms)	**Total** waiting Time (ms)
Shortest Remaining Time First	79	48
Round Robin (quantum = 2)	129	98
Priority (Preemptive)	106	75

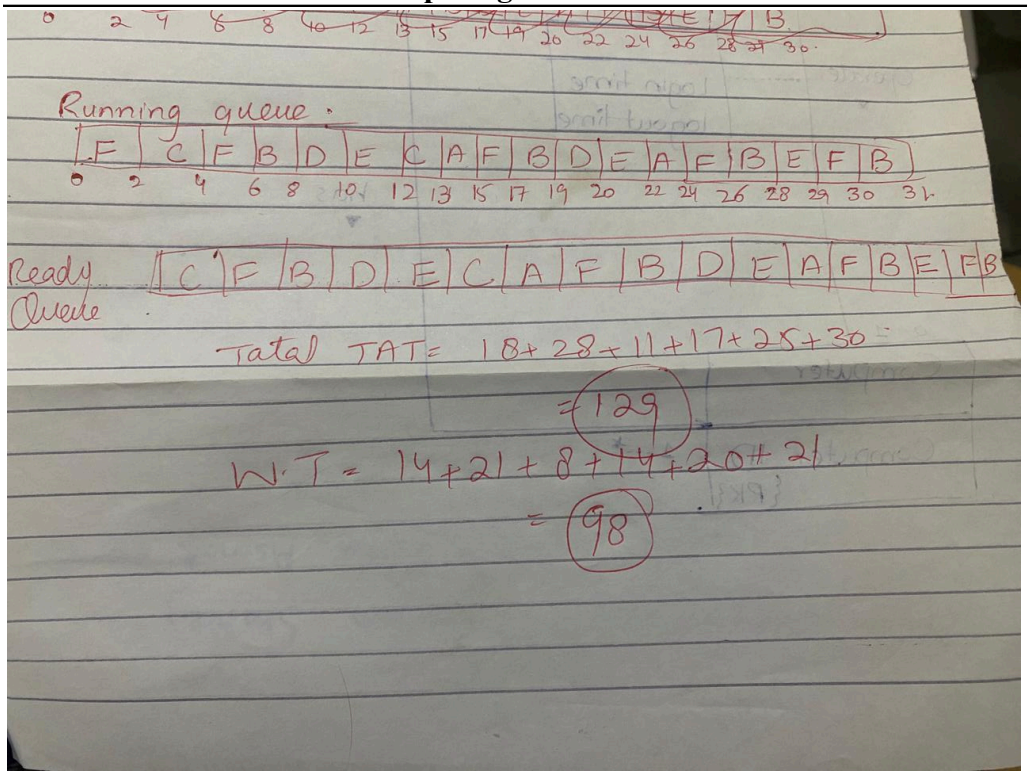
{**3 [+-1] OR 2 if chart is ok but value is wrong OR ZERO** marks for each calculation }
(give 2 marks if students answer is +-5 to this answer)

Show calculations for the above question in the space below (MANDATORY):

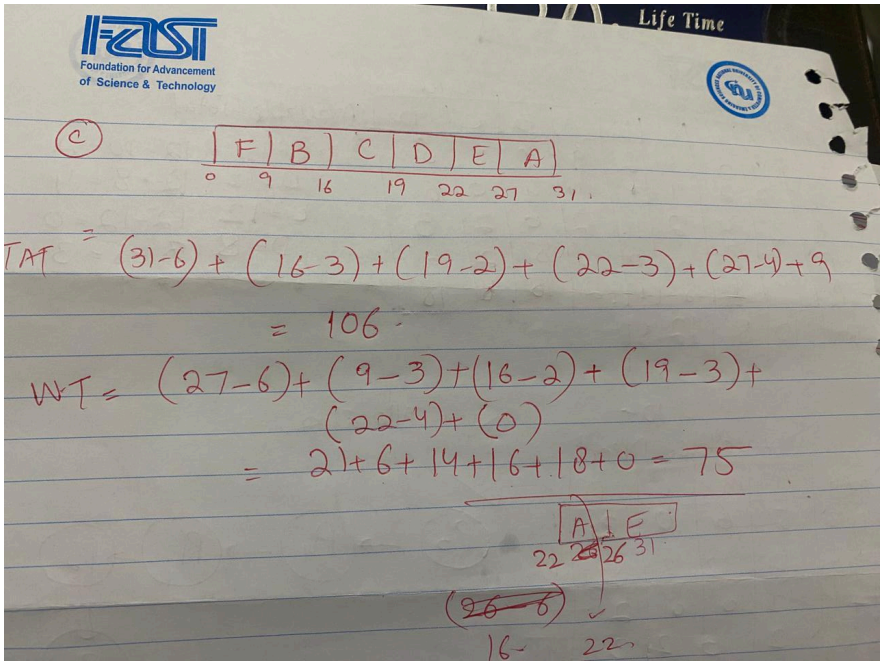
Shortest Remaining Time First



Round Robin (quantum = 2)



Priority (Preemptive)



b). Schedule the following processes using ***Multi-level queue scheduling*** algorithm. Assume **Queue1** has *time quantum* of *4 ms* and **Queue2** has *time quantum* for *3 ms*.

Process	Arrival Time	CPU burst time (ms)	Arrival Queue
A	0	10	1
B	2	7	2
C	4	6	1
D	8	5	2
E	12	8	2
F	13	7	1

Keep the following things in mind while answering the questions below: [7 marks]

- Queue1 has higher priority than Queue2.
- The arrival queue for the new process is also provided in the table.
- Show you working in the given space. No marks will be given for the direct answers.
- **Your answer should contain numbers only (i.e. time in terms of ms)**

Question	Answer
What will be the ***response time*** for **Process E**? **[2 marks]**	**17**
What will be the ***waiting time*** for **Process C**? **[2 marks]**	**4**
What will be the ***turnaround time*** for **Process F**? **[3 marks]**	**10**

Show the working for the above question in the space below (MANDATORY):

{**marks will be given if the working matches the answer above**}

Gantt chart

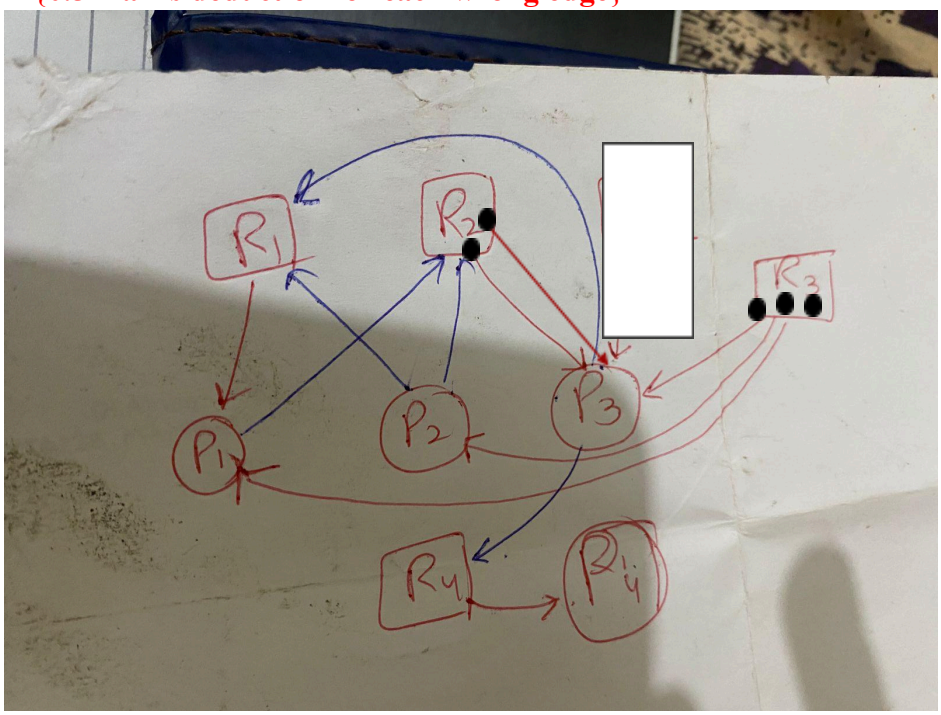

A C A C A F F B D E B D E B E
0 4 8 12 14 16 20 26 21 Gantt chart 35 29 40 41 43

Question 3 [10 Marks]

A system has three processes (P1, P2, P3, P4) and four reusable resources (R1, R2, R3, R4). There is one instance of R1, two instances of R2, three instances of R3 and one instance of R4. P1 holds an R1 and an R3 and is requesting an R2. P2 holds an R3 and is requesting an R1 and an R2. P3 holds two R2 and an R3 and is requesting an R1 as well as an R4. P4 holds R4.

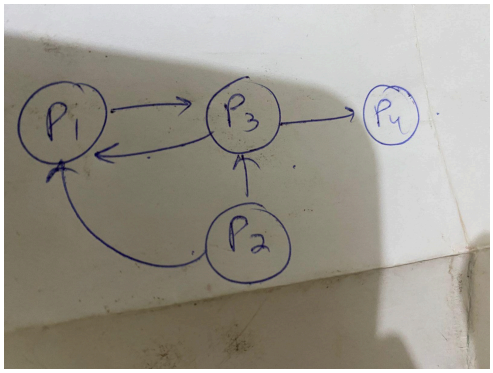
a). Draw the resource allocation graph for this system. Use the style of diagram used in the lecture slides. [4 marks]

{**0.5 marks deduction for each wrong edge**}



b). Create Wait-for graph for the system**. [3]**

{**3 or ZERO**}



c). Is this system deadlocked? If so, state which processes are involved. **[3 marks]**

{**1** marks for Yes/No + **2** for Processes}

Yes, P1 and P3 involved (cycle)

Question 4 [10 Marks]

Suppose we have the following resources:

- 5 processes P1 through P5
- 4 resource types A (12 instances), B (12 instances), C (12 instances) and D (8 instances)

Below Snapshot is taken at time T0

	Allocation				Max. Request			
	A	B	**C**	**D**	**A**	**B**	**C**	**D**
P1	4	0	4	2	9	6	4	6
P2	2	2	1	1	4	4	3	6
P3	0	0	2	1	2	0	2	3
P4	4	2	0	0	6	5	2	3
P5	0	4	5	1	2	10	8	10

a). Calculate available vector: [2 marks]

Available			
A	B	**C**	**D**
2	4	0	3

{0.5 marks for each correct entry}

b). Calculate Need Matrix: [2 marks]

	Need			
	A	B	**C**	**D**
P1	5	6	0	4
P2	2	2	2	5
P3	2	0	0	2
P4	2	3	2	3
P5	2	6	3	9

{0.5 marks for each correct row}

c). Suppose at time T0, Process P2 requests (1,2,0,4) from its Max need. Should this request be granted? Why or why not? **[2 marks]**

{**0.5** marks for Yes/No answer + **1.5 marks** for correct reason with calculation}

No ,

Reason : P2 requests (1,2,0,4) but available are (2,4,0,3) , therefore 4 D resources are required but, 3 are available.

d). Consider the provided snapshot, determine the safe sequence. No marks will be provided without showing proper working in the space provided below: [4 marks]

{**2** marks for solution + **2** for calculations}

(Also give marks if student has done calculation without drawing table)

Sequence: P3 → P4 → P1 → P2 , at P5 deadlock occurs

PROCESSES	A	B	C	D
INTIALLY	2	4	0	3
P3	2	4	2	4
P4	6	6	2	4
P1	10	6	6	6
P2	12	8	7	7

Question 5 [20 Marks]

Consider a **32-bit** logical address size where page size is allocated **16 Kbits**. Given Size of physical memory is **1Gbits**. Suppose that you decide to use a single-level page table.
Answer the following questions: (All answers must be accompanied with correct and complete calculations)

a. How many bits would represent page number in logical address space? [2 marks]

Size of logical address space = 2^m
# of pages (2^{m-n}) × page size (2^n)

m=32
n = $\log_2$ (page size) = $\log_2$ (16 x 1024) = $\log_2$(16384) = 14

m-n = 32 - 14 = **18**
{**1** marks for formula + **1** for correct result}

b. How many bits would represent an offset within a page in logical address space? [2 marks]

n = **14**
{**2 marks or ZERO**}

c. How many total pages would be created in logical address space? [2 marks]

# of pages (2^{m-n}) = $\mathbf{2^{18}}$ = **262,144**

…. {Any of the two bold values will be accepted, **2 marks** or **ZERO**}

d. How many total frames would be created in physical address space. [2 marks]

Physical memory size = 2^m = 1 Gbits (1024 Mbits = $\mathbf{2^{30}}$ bits)
Page Size = $2^n = 2^{14}$
# of frames (2^{m-n}) = 2^{m-n} = $\mathbf{2^{30}-2^{14} = 2^{16} = 65536}$
{**1** marks for correct 2^m + 1 marks for formula }

e. How much maximum memory is needed to store the page table? [4 marks]

Total memory = Number of entries * Size of each entry
Total memory = 2^16 entries * 16 bits/entry
Total memory = **2^22 bits**
Total memory = 2^22 bits / 8 bits/byte
Total memory = **2^20 bytes = 1,048,576 bytes**

Total memory = **1 MB (approx)**
{Any of above 4 bold values will be accepted, **4 marks or ZERO**}

f. Assume the access time for main memory is 100 ns and the page fault service time is 10ms. Assume the page table is stored in main memory.

i). If no TLB is used, how long does it take for a user program to load a byte from main memory into a CPU register? Assume the desired page is already in the main memory. [4 marks]
{**1** marks for formula + **3** for correct result}

Effective Memory Access Time (EMAT) = Page table access time + memory access time
= 100 ns + 100 ns
= **200 ns**

ii). Suppose we add a TLB to the system to speed up logical-to-physical address translation. With a TLB access time of 10 ns and a TLB hit rate of 0.9, what is the effective memory access time? Assume we only do a main-memory page table lookup after a TLB miss, and that the desired pages are always in main memory. [4 marks]
{**2** marks for formula + **2** for correct result}

EMAT = (TLB hit rate * time for TLB hit) + (1 - TLB hit rate) * (time for TLB miss)

Total time for TLB hit = 10 ns + 100 ns = 110 ns
Total time for TLB miss = 10 ns + 100 ns + 100 ns = 210 ns

EMAT = (0.9 * 110 ns) + (1 - 0.9) * 210 ns
EMAT = 99 ns + 21 ns
EMAT = **120 ns**

Question 6 [20 Marks]

Consider *Dynamic Partitioning* using Linked Lists, if the processes arrive in the following sequence:

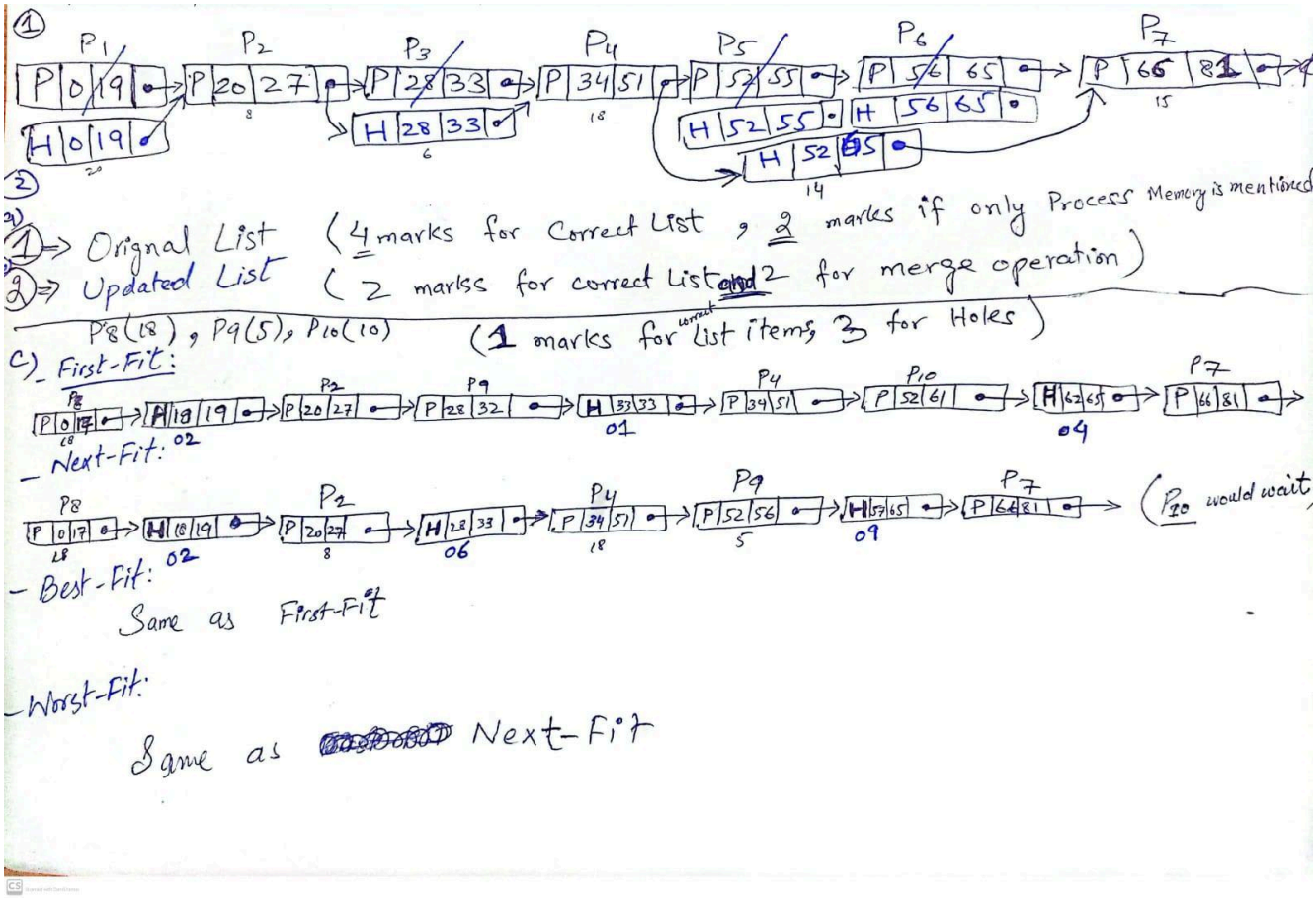
Process	Memory Requirement (KB)
P1	20
P2	8
P3	6
P4	18
P5	4
P6	10
P7	15

a). Create the linked list. [4 marks]

b). Show the updated linked list if the processes P1, P3, P5, and P6 leave the system in the sequence. [4 marks]

c). Suppose after the above processing, now, we want to Fit the upcoming process P8, P9 and P10 with memory requirements of the processes 18, 5, 10 respectively. How would each of the first-fit, next-fit, best-fit, and worst-fit algorithms. [3 * 4 = 12]

Solution:



Question 7 [10 Marks]

a). Suppose a system in which there are 2 types of processes, type A and type B processes. All processes of type A execute the same code, and all processes of type B execute the same code. The code for each process type is shown below.

Process A	Process B
P(X) V(Y)	P(Y) P(Y) V(X) Y(Y)

Here X,Y are general semaphores. X is initialized to 2, and Y is initialized to 0. Suppose three processes of type A and 2 processes of type B are brought in execution simultaneously. Answer the following questions.

i). Is it possible for processes to finish in the order of **AABAB**? If so, show an execution sequence that results in this order. If not, explain why as accurately as possible.**[5 marks]**

2 Marks for writing possible
3 Marks for solving it
possible

A_1	A_2	B_1	A_3	B_2	*Semaphore X*	*Semaphore Y*
					2	0
P(X)					1	0
V(Y)					1	1
	P(X)				0	1
	V(Y)				0	2
		P(Y)			0	1
		P(Y)			0	0
		V(X)			1	0
		V(Y)			1	1
			P(X)		0	1
			V(Y)		0	2
				P(Y)	0	1
				P(Y)	0	0
				V(X)	1	0
				V(Y)	1	1

ii). Is it possible for processes to finish in the order **AABBA**? If so, show an execution sequence that results in this order. If not, explain why as accurately as possible.**[5 marks]**

2 MArks for writing not possible
3 marks for reason

The sequence *AABBA* is impossible. From the above execution sequence, after *AA* semaphores *X* and *Y* have counters 0 and 2. If *B* follows, *X* and *Y* will be 1 and 1. Because *Y*'s counter is 1, the next *B* can only pass its first `P(Y)` and blocks by the second `P(Y)`. Then, the second *B* will be blocked by its first `P(Y)`, and, as a result, the order of *AABBA* is impossible.

b). Given The code below, we want to guarantee that a1 happens before b2 and b1 happens before a2. You can use upto 2 semaphores in writing your solution. Be sure to specify the names and initial values of your semaphores. This synchronization problem has a name; it's a rendezvous. The idea is that two threads rendezvous at a point of execution, and neither is allowed to proceed until both have arrived. **[5 marks]**

Thread A

```
1 statement a1
2 statement a2
```

Thread B

```
1 statement b1
2 statement b2
```

2 marks for two semaphores
(1 for semaphore declaration + 1 for initialization)

Semaphore aArrived(0)
Semaphore bArrived(0)

Thread A

```
1 statement a1
2 bArrived.wait()
3 aArrived.signal()
4 statement a2
```

Thread B

```
1 statement b1
2 bArrived.signal()
3 aArrived.wait()
4 statement b2
```

1.5 marks for Thread A
(
0.5 marks for statements +
0.5 marks for **semaphore2.Wait()**
0.5 marks for **semaphore1.Signal()**
)

1.5 marks for Thread B
(
0.5 marks for statements +
0.5 marks for **semaphore2.Signal()**
0.5 marks for **semaphore1.Wait()**
)