THE HONG KONG POLYTECHNIC UNIVERSITY

DEPARTMENT OF COMPUTING

SEMESTER 2023 / 2024

PROGRAMME : 2011, 2012, 232, 241

YEAR : 2023/2024

SUBJECT CODE : COMP2432

SUBJECT TITLE : OPERATING SYSTEMS

TIME ALLOWED : 2 HOURS

THIS QUESTION PAPER HAS 8 PAGES (ATTACHMENTS INCLUDED)

INSTRUCTIONS TO CANDIDATES:

1. THIS MID-TERM TEST CONSTITUTES 20% OF THE GRADE.
   ANSWER ALL QUESTIONS. THE TIME ALLOWED IS 120 MINUTES.
2. ANSWER ALL QUESTIONS NEATLY AND CLEARLY IN THE ANSWER BOOKLET
3. OVERWRITTEN AND SCRIBBLED ANSWERS WILL NOT BE CONSIDERED FOR EVALUATION
4. ANY ELECTRONIC DEVICES OR CALCULATORS ARE NOT PERMITTED TO BE USED DURING THE EXAMINATION
5. ONLY 1 PAGE (2 SIDES) A4 SIZE CHEAT SHEET IS PERMITTED.
6. THE CHEAT SHEET SHOULD HAVE STUDENT LAST NAME, FIRST NAME AND STUDENT ID CLEARLY WRITTEN ON THE TOP OF THE CHEAT SHEET.

NOTES TO GENERAL OFFICE:
(E.G. STATIONERY AND NO. OF ANSWER BOOKS REQUIRED)

Section A: Multiple Choice Questions

Question 1 [3 **Marks**]

Choose the right order of arrangement for the following types and commands in the layered architecture (similar to OSI model) of the Linux;

a. CPU, b. ksh, c. bash, d. ls, e. grep, f. spreadsheet

A. (c, d, e, f, a, b) B. (b, c, d, e, f, a) C. (a, f, b, c, d, e) D. (f, b, c, d, e, a)

Question 2 [3 **Marks**]

There are 4 tasks that requires to execute by using **synchronous** and **asynchronous** processing. The total execution time taken by these tasks by using these two methods are;

Task 1 requires 020 milliseconds
Task 2 requires 005 milliseconds
Task 3 requires 003 milliseconds
Task 4 requires 100 milliseconds

A. (28, 100) B. (100, 123) C. (128, 100) D. (100, 128)

Question 3 [3 **Marks**]

Consider the following code snippet using the *fork()* and *wait()* system calls. Assume that the code compiles and runs correctly, and that the system calls run successfully without any errors. The total number of times the *printf()* statement is executed is _____.

A. 8 B. 10 C. 14 D. 16

```
int x =3;
while(x > 0){
fork ();
printf("hello");
wait (NULL) ;
X-- ;
}
```

Question 4 [3 **Marks**]

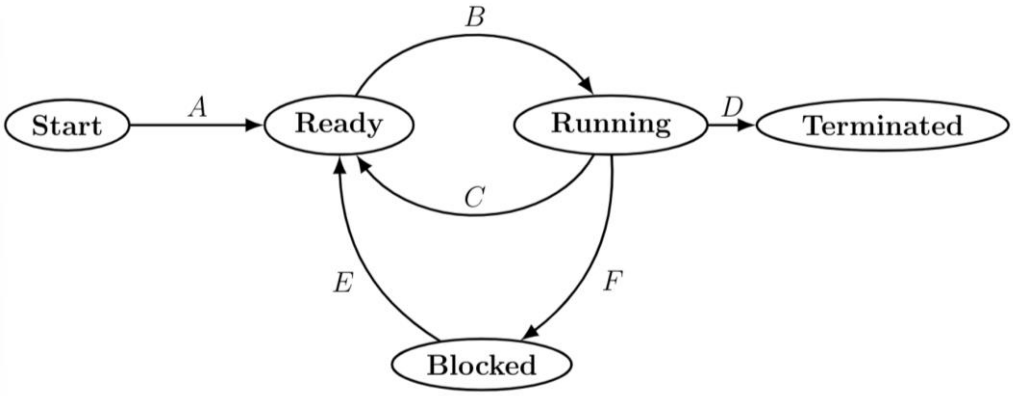
In the following process state transition diagram for a uniprocessor system, assume that there are always some processes in the ready state:

Now consider the following statements:

I. If a process makes a transition (leave) ***D***, it would result in another process making transition ***A*** immediately.
II. A process P_2 in blocked state can make transition ***E*** while another process P_1 is in running state.
III. The OS uses preemptive scheduling.
IV. The OS uses non-preemptive scheduling.

Which of the above statements are TRUE?

A. I and II
B. I and III
C. II and III
D. II and IV



Question 5 [**3 Marks**]

Which of the following scheduling algorithms is non-preemptive?

A. Round Robin
B. First-In First-Out
C. Multilevel Queue Scheduling
D. Multilevel Queue Scheduling with Feedback

Question 6 **[3 Marks]**

Consider the following code snippet using the *pipe()* system calls. Assuming the below fragment with child sending data to parent; fill the 4 missing blanks with right option

A. (1, 0, 0, 1) B. (0, 1, 1, 0) C. (1, 0, 1, 0) D. (0, 0, 1, 1)

```
if (pipe(fd) < 0) { printf("Pipe creation error\n"); exit(1); }
pid = fork();
if (pid < 0) { printf("Fork failed\n"); exit(1); }
else if (pid == 0)
   close(fd[_____]);
   //program here ...
   write write write
   close(fd[_____]);
} else
   close(fd[_____]);
   //program here ...
   read read read
   close(fd[_____]);
}
```

Prog. 1

Question 7 **[3 Marks]**

Consider the following table of arrival time and burst time for three processes P_0, P_1 and P_2. The pre-emptive shortest job first scheduling algorithm is used. Scheduling is carried out only at arrival or completion of processes. What is the average waiting time for the three processes?

Process	Arrival Time	Burst Time
P_0	0 ms	9 ms
P_1	1 ms	4 ms
P_2	2 ms	9 ms

A. 5.0 ms B. 4.33 ms C. 6.33 ms D. 7.33 ms

Question 8 [**3 Marks**]

Consider the following code snippet implementing the *mkfifo()* function.

What should be replaced in the code to give *mkfifo()* full access privilege to owner, R,W access privilege to group and R, X privilege access to others

A. 0765 B. 0567 C. 0467 D. 0756

```
if (mkfifo(pipename,0600) < 0) {
  printf("Pipe creation error\n");   exit(1); }
returnpid = fork();
if (returnpid < 0) {
  printf("Fork failed\n"); exit(1);
} else if (returnpid == 0) {
if ((fd = open(myworld, O_RDONLY)) < 0)
{
   printf("Pipe open error\n"); exit(1); }
  read read read
  exit(0);
}
```

Question 9 [**3 Marks**]

The command **od -c -x lab1.pdf | more** outputs hexadecimal number in the form of

A. Little Endian B. Big Endian C. 4-Bytes D. 8-Bytes

Question 10 [**3 Marks**]

Choose the TRUE statements

a. Monolithic kernel allows IPC/File Systems to work under kernel space
b. UNIX is an example of Monolithic kernel
c. Monolithic kernel is also known as Darwin Kernel
d. Microkernel separates most of the OS services in user space

A. a, b, c B. a, b, d C. b, c, d D. All of the above

Cont'd…

Section B: Fill in the Blanks Questions

Question 11 [**5 Marks**]

Consider a machine with the scenario of CPU scheduling where logical address space is defined as 4GB with page size of 4KB and physical address space of 64MB. What would be the number of pages and number of frames?

A. No. of Pages _____ B. No. of Frames _____

Question 12 [**5 Marks**]

Consider a memory management system that uses a page size of 2KB. Assume that both the physical and virtual addresses start from 0. Assume that the pages 0, 1, 2 and 3 are stored in the page frames 1, 3, 2 and 0 respectively. The physical address **base register** value of page 1 is _____

Question 13 [**5 Marks**]

Consider a paging scheme is using TLB where the TLB access time is 10 ns and main memory access time takes 50 ns. What is effective memory access time (in ns) if TLB hit ratio is 90% and there is no page fault.

Question 14 [**5 Marks**]

Consider the following table of arrival time and burst time for four processes P_1 to P_4. The Round Robin scheduling algorithm with time quantum 2 is used. Scheduling is carried out only at arrival or completion of processes quantum. Draw the **Gantt Chart** and write is **total** number of times **context switching** take places until all the processes terminate?

Process	Arrival Time	Burst Time
P_1	0 ms	5 ms
P_2	1 ms	4 ms
P_3	2 ms	2 ms
P_4	4 ms	1 ms

Cont'd…

Section C: Descriptive & Programming Questions

Question 15 [**15 Marks**]

Assume that the following processes are the only processes in a computer system and that there are no input/output requests from all the given processes. Given the following arrival time and CPU burst time for each process, draw the **Gantt chart** and compute the **waiting time** and **turnaround time** for each process by filling in the table in answer booklet for the following CPU Scheduling algorithms:

a. Shortest Response Time First

b. Round Robin with time quantum = 4

Process	Arrival Time	CPU Burst Time
P_1	0	10
P_2	2	6
P_3	3	1
P_4	5	3

Question 16 [**15 Marks**]

Consider five memory partitions of size 100 KB, 500 KB, 200 KB, 450 KB and 600 KB in same order. If sequence of requests for blocks of size 212 KB, 417 KB, 112 KB and 426 KB in same order come, then which of the following algorithm makes the efficient use of memory?

A. For each algorithm draw the memory table/frames and discuss the efficient algorithm.
B. Show how you could satisfy (memory utilization) of all requests if you see all of them in advance.

A. Best fit algorithm,

B. First fit algorithm

C. Worst fit algorithm

Cont'd…

Answer Question 17 and Question 18 in 300 – 500 words ONLY

Question 17 Answer any ONE **[10 Marks]**

A. List and explain the functionality of different types of OS with examples. **OR**

B. Distinguish between a Zombie and an Orphan process. In Linux, orphan processes are *adopted* by a special process. *Who* is that special process?

Question 18 Answer any ONE **[10 Marks]**

A. What is the need for paging? Distinguish between paging and segmentation. In memory protection what is the page-table length register is used for? **OR**

B. Consider a computer system with 8KB main memory and frame size of 256 bytes, with a logical address length of 12 bits. The logical address 010011010010 is generated by the CPU for a user process, with Page Table Base Register value of 0101010101010. Given the page table content below, **determine** the physical address in the main memory for the CPU generated logical address 010011010010. Working steps are optional.

Page	Frame Number
0	10001
1	00010
2	10110
3	00111
4	10011
5	00100

```
pipe(fd1); pipe(fd2); p = fork();
if (p > 0) strcpy(buf,"hello"); else strcpy(buf,"bye");
if (p > 0) write(fd1[1],buf,strlen(buf)); else write(fd2[1],buf,strlen(buf));
if (p > 0) { n = read(fd2[0],buf2,80); buf2[n] = 0; }
if (p > 0) p = fork();
if (p > 0) strcpy(buf,"welcome");
if (p > 0) write(fd2[1],buf,strlen(buf)); else write(fd1[1],buf,strlen(buf));
if (p > 0) { n = read(fd1[0],buf,80); buf[n] = 0; printf("%s\n",buf); }
```

Question 19 – BONUS **[5 Marks]**

Write a **program** fragment demonstrating a linear communication IPC model between a parent, child and a grandchild. Follow the sequence of communication in the order of parent writing to child and child writing to grandchild. You should clearly **mark** the reading and writing ends in your program and **draw** the clear diagrams depicting the file descriptors open and closed ends.

Question 20 – BONUS **[5 Marks]**

Assuming that other statements (in above program) are correctly written (e.g., definition of variables), **how many** processes are created altogether? **How many** possible outputs can there be when the following program fragment is executed? **List** out those possible outputs.

Note: *BONUS points will be counted only if the entire solution will be correct. No partial points.*

*** End ***

The Hong Kong Polytechnic University (PolyU)
2023-2024 Academic Year Semester II
COMP 2432 Operating Systems

Answer Sheet

SEAT NO: _____

Student Last Name: _____

Student First Name: _____

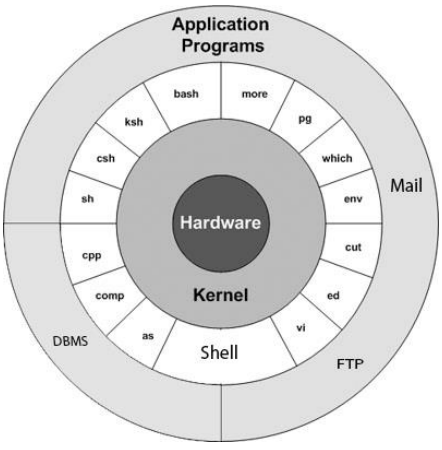
Student ID: _____

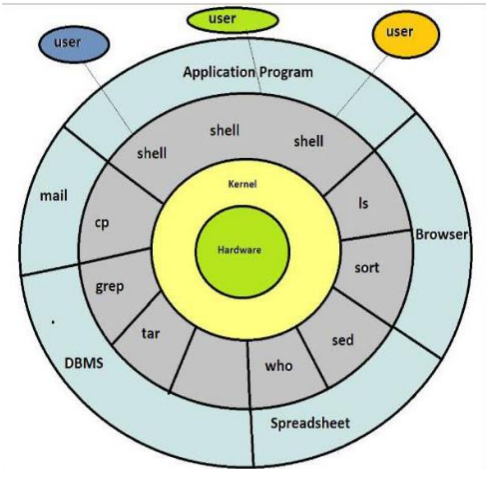
Total Marks:

Section A (Question 1-10: 3 marks) **Section B** (Question 11-14: 5 marks)

Question	1	2	3	4	5	6	7	8	9	10
Answer	D	C	C	C	B	B	A	A	A	B
Question	11A	11B	12	13						
Answer	2^{20}	2^{14}	6144 or 6KB	65 ns						

Question 1



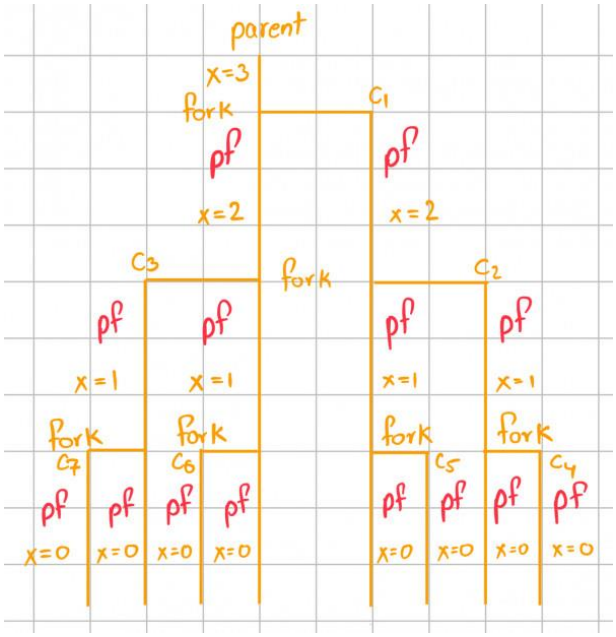


Student Last Name: _____ Student First Name: _____

Student ID: _____ SEAT NO: _____

Question 3

The printf statement is executed 14 times. The code repeatedly forks processes inside a while loop. Each iteration doubles the number of processes. Initially, one process prints “hello”. After each fork, both the parent and child processes print “hello”. The loop iterates three times, resulting in 2^3 = 8 processes. Additionally, two processes from the previous iteration wait for each other before proceeding. Thus, the total count is 8 + 4 + 2 = 14 “hello” messages printed before the loop terminates.



In first iteration, parent will spawn a child c1, both will have x=3 and execute printf statement 3 times.

In second iteration, parent and child c1 will spawn 2 children c2 and c3, both children will have x=2 and execute printf statement 2 times.

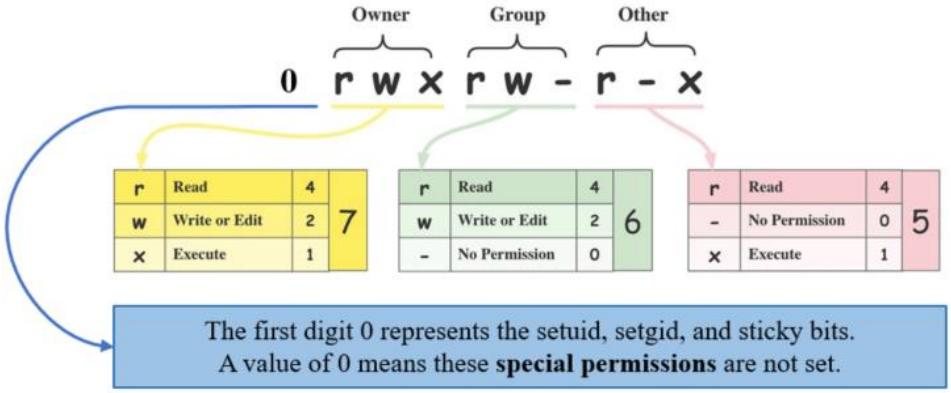
In third iteration, parent and children c1, c2, c3 will spawn 4 children c4, c5, c6 and c7, all 4 children will have x=1 and execute printf statement only once.

Loop terminates in all processes. Number of times print statement is executed is
3*2+2*2=4*1=14

Question 8

Linux File Permissions

Binary	Octal	String Representation	Permissions
000	0 (0+0+0)	---	No Permission
001	1 (0+0+1)	--x	Execute
010	2 (0+2+0)	-w-	Write
011	3 (0+2+1)	-wx	Write + Execute
100	4 (4+0+0)	r--	Read
101	5 (4+0+1)	r-x	Read + Execute
110	6 (4+2+0)	rw-	Read + Write
111	7 (4+2+1)	rwx	Read + Write + Execute



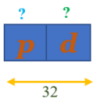
Question 11

- Given:
- Logical address space = 4 GB
- Physical address space = 64 MB
- Page size = 4 KB

Step 1: Make binary approximations

*(**DO NOT** convert the Byte as CPU is Byte addressable)*

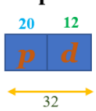
LAS = 4 GB = $2^2 * 2^{30} = 2^{32}$

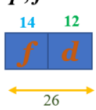


PAS = 64 MB = $2^6 * 2^{20} = 2^{26}$

Page Size = 4 KB = $2^2 * 2^{10} = 2^{12}$

Step 2: Calculate *p*, *f* and *d*





- No. of Pages = ?
- No. of Frames = ?
- No. of entries in page table = ?
- Size of page table = ?

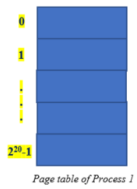
Step 3: Calculate No. of pages and No. of Frames

Page number bits are **20**. The number of pages are 2^{20}.

Frame number bits are **14**. The number of frames are 2^{14}.

Step 4: No. of entries in page table

No. of entries in page table =
No. of pages in a process
= 2^{20}



Page table of Process 1

Step 5: Size of page table

Since page table contains a frame number and each frame number here is represented in 14 bits.

The number of entries in page table = 2^{20} x 14 bits

$2^1 = 2$	$2^9 = 512$
$2^2 = 4$	$2^{10} = 1024$
$2^3 = 8$	$2^{11} = 2048$
$2^4 = 16$	$2^{12} = 4096$
$2^5 = 32$	$2^{13} = 8192$
$2^6 = 64$	$2^{14} = 16384$
$2^7 = 128$	$2^{15} = 32768$
$2^8 = 256$	$2^{16} = 65536$

Abbr.	Prefix name	Binary approximation
K	kilo-	$1,024 = 2^{10}$
M	mega-	$1,024^2 = 2^{20}$
G	giga-	$1,024^3 = 2^{30}$
T	tera-	$1,024^4 = 2^{40}$
P	peta-	$1,024^5 = 2^{50}$
E	exa-	$1,024^6 = 2^{60}$

Note 1: Always Page size = frame size
Note 2: CPU is Byte addressable

Question 12

Page 1 is stored in Main Memory frame 3. Given that the page size is 2KB, the starting address of each page frame is simply the page frame number multiplied by the page size.

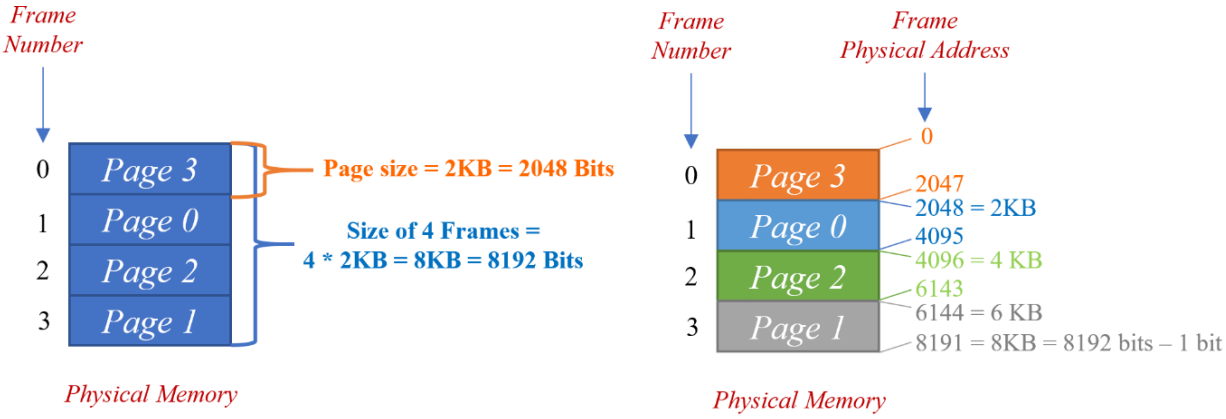
So, the physical address or starting address of page 1 is:

Page frame number *Page size* = *3** 2KB = 6KB

The ending address of page 1 is the starting address plus the page size, minus one (since we start counting from 0). So, the ending address of page 1 is:

Starting address + Page size - 1 = 6KB + 2KB - 1 = 8KB - 1 = 7.999KB

So, the physical address or starting address of page 1 is 6KB, and the ending address is 7.999KB.

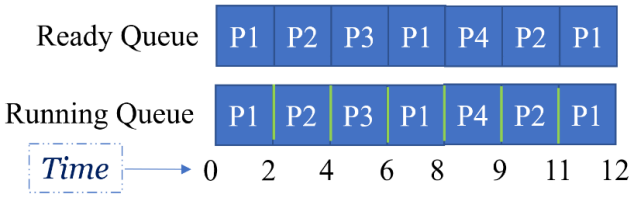


Question 13. (5 marks)

cache access time + (2-h) * memory access time = 10 + (2 - 0.9) * 50 = 65ns

Question 14. (5 marks)

Gantt Chart:



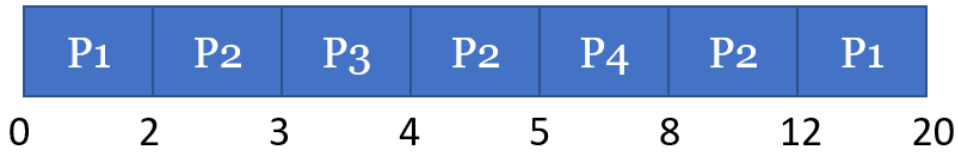
Total Number of contest switches: **6**

Student Last Name: _____ Student First Name: _____

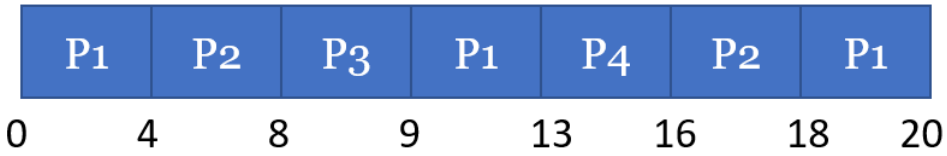
Student ID: _____ SEAT NO: _____

Section C (Question 15, 16: 15 marks, Question 17: 20 marks)

Gantt Chart (SRT):



Gantt Chart (RR):



			a. (SRT)		*b. (RR)*	
Process	**Arrival Time**	**CPU Burst Time**	**Waiting Time**	**Turn Around Time**	**Waiting Time**	**Turn Around Time**
P_1	0	10	**10**	**20**	**10**	**20**
P_2	2	6	**4**	**10**	**10**	**16**
P_3	3	1	**0**	**1**	**5**	**6**
P_4	5	3	**0**	**3**	**8**	**11**

Question 16 (15 marks)

Best Fit: Memory wastage: 100 + (500-417)+(200)+(450-324)+(600-426) = 683 KB

First Fit: Memory wastage: 100 + (500-324)+200+(450-417)+(600-426) = 683 KB

Worst Fit: Memory wastage: 100 + (500-417)+200+(450-426)+(600-324) = 683 KB

Reasoning: First fit is the most efficient algorithm.

The efficiency of memory allocation algorithms depends on the specific requirements and the sequence of memory requests. In this case, we have five memory partitions and four memory requests. Let's analyze each algorithm:

A**. Best Fit Algorithm:**

The best fit algorithm searches the entire list and finds the smallest block that is large enough to accommodate the request. It aims to minimize the leftover space.

For the given sequence of requests, the best fit algorithm would allocate memory as follows:

- 212 KB request is allocated to the 500 KB partition (leaving 288 KB free)
- 417 KB request is allocated to the 450 KB partition (leaving 33 KB free)
- 112 KB request is allocated to the 200 KB partition (leaving 88 KB free)
- 426 KB request is allocated to the 600 KB partition (leaving 174 KB free)

The total leftover space is 288 + 33 + 88 + 174 + 100 (unused 100 KB partition) = 683 KB.

B**. First Fit Algorithm:**

The first fit algorithm searches the list and allocates memory from the first block that is large enough to accommodate the request.

For the given sequence of requests, the first fit algorithm would allocate memory as follows:

- 212 KB request is allocated to the 500 KB partition (leaving 288 KB free)
- 417 KB request is allocated to the 450 KB partition (leaving 33 KB free)
- 112 KB request is allocated to the remaining space in the 500 KB partition (leaving 176 KB free)
- 426 KB request is allocated to the 600 KB partition (leaving 174 KB free)

The total leftover space is 176 (remaining in 500 KB partition) + 33 (450-417) + 174 (600-426) + 200 (unused 200 KB partition) + 100 (unused 100 KB partition) = 683 KB.

Student Last Name: _____ Student First Name: _____

Student ID: _____ SEAT NO: _____

C. **Worst Fit Algorithm:**

The worst fit algorithm searches the entire list and allocates memory from the largest block.

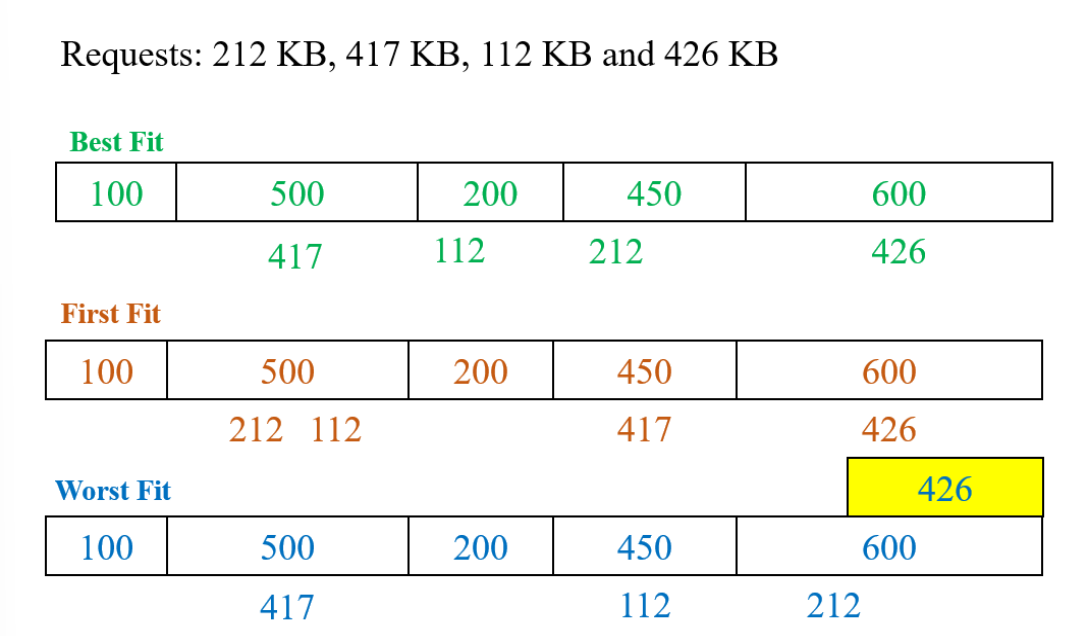
For the given sequence of requests, the worst fit algorithm would allocate memory as follows:

- 212 KB request is allocated to the 600 KB partition (leaving 388 KB free)
- 417 KB request is allocated to the 500 KB partition (leaving 83 KB free)
- 112 KB request is allocated to the 450 KB partition (leaving 388 KB free)
- 426 KB request is in the waiting

The total leftover space is 388 + 83 + 338 + 300 (unused partition) = 1109 KB.

In conclusion, all two algorithms - Best Fit, First Fit - have the same memory utilization in this specific case. However, the First Fit algorithm is generally preferred when the memory utilization is the same. This is because the First Fit algorithm has a lower computational overhead compared to the Best Fit and Worst Fit algorithms.

The First Fit algorithm stops searching as soon as it finds a free block that can accommodate the memory request, while the Best Fit and Worst Fit algorithms need to search the entire memory to find the smallest or largest free block, respectively. Therefore, in terms of time complexity and efficiency, the First Fit algorithm would be the most efficient choice in this scenario.



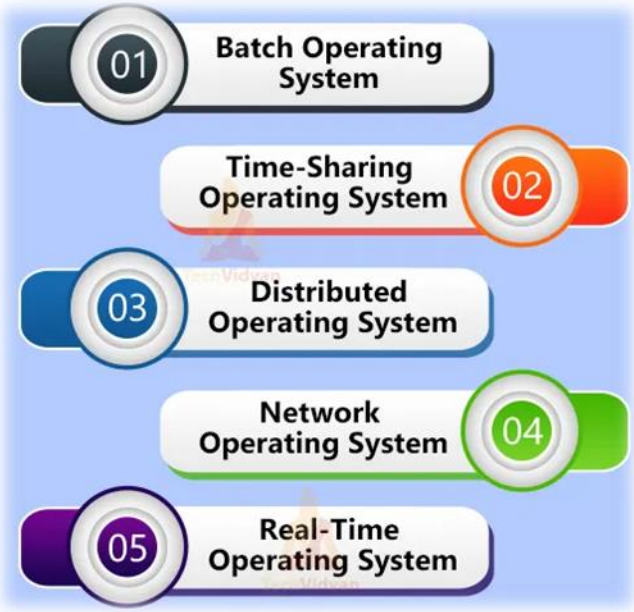
Student Last Name: _____ **Student First Name:** _____

Student ID: _____ **SEAT NO:** _____

Section C (Question 17 & Question 18. (20 marks)

Types of Operating Systems

- Batch processing
  - A job is a sequence of programs represented by an input data sets (often a deck of cards).
  - One job is executed one at a time and programs within each job are executed sequentially.
  - Common systems in 60's like IBM JCL-related (job control language) systems.
- Multiprogramming
  - Programs are executed in interleaved manner by the CPU.
  - CPU is given to another program when the current program is waiting for I/O.
  - Common systems in 70's and Unix.
- Time-sharing
  - Processes are executed in interleaved manner by the CPU.
  - Same as multiprogramming, but system is interactive, so each process must get CPU without waiting for too long.
  - e.g. Unix, Linux, Windows Server
- Real-time
  - A time-sharing system where processes have completion deadlines that must be met.
  - Hard real-time systems have firm deadlines.
    - Missing a deadline makes a process useless, e.g. examination.
  - Soft real-time systems have soft deadlines.
    - Missing a deadline reduces value of a process, e.g. assignment.
  - Important in high-valued and mission critical applications.
    - Banking transactions, airline reservation, space shuttle control.
- Distributed
  - Operated upon a collection of computers scattered across the network.
  - Advanced from network operating systems.
  - Need to coordinate the share of resource and execution of processes to achieve a common goal.
  - Examples:
    - Client/server system, internet computing systems.



Type	Definition	Example of Use
Batch OS	Data or programs are collected grouped and processed at a later date.	Payroll, stock control and billing systems.
Interactive OS	Allows the user and the computer to be in direct two-way communication	Select from a menu at ATM.
Real-time OS	Inputs immediately affect the outputs. Timing is critical	Control of nuclear power plants, air traffic control systems.
Network OS	Allow a computer on a network to serve requests from other computers for data and provide access to other resources such as printer and file systems.	Manage simultaneous access by multiple users
Multiuser OS	Handle many people running their programmes on the computer at the same time	A number of terminals communicating with a central computer which allocates processing time to each terminal in turn.
Multiprogramming OS	Ability to run many programmes apparently at the same time.	Mainframe systems. Each job is allocated a small amount of processing time (*time slice*) in turn.
Multitasking OS	The ability to hold several programmes in RAM at one time but the user switches between them.	Usually uses GUI's. Facilitates import and export of data.

17B. Zombie and Orphan processes are two types of special processes in Unix-like operating systems.

Zombie Process: A Zombie process is a process that has completed execution but still has an entry in the process table. This is a dead process that is not doing any work but still has a process table entry. This usually happens when the parent process doesn't read the exit status of its child, leaving the child in a "zombie" state.

Orphan Process: An Orphan process is a running process whose parent process has finished or terminated. Though their parent process has ended, these processes continue to run.

In Linux, when a parent process dies before its child processes finish, those child processes become orphaned. These orphan processes are adopted by a special process known as the "init" process. The init process is the first process started during system boot, with a process ID (PID) of 1, and it runs until the system shuts down. Its main task is to adopt orphaned processes and to clean up after them when they terminate, preventing them from becoming zombie processes.

18A. Paging is needed for several reasons:

1. Allows a big process to be divided into multiple pages for memory allocation
2. It allows the physical address space of a process to be noncontiguous, which makes the process of swapping a process in and out of main memory more efficient.
3. It simplifies memory management by allowing the operating system to deal with fixed-size pages rather than variable-sized processes.
4. It allows the use of virtual memory, where a process can be larger than physical memory. The parts of the process that are not currently in use can be swapped out to disk, freeing up physical memory.

Paging and Segmentation are both memory management schemes. Paging divides the computer's primary memory into fixed-size units called page frames, and the program's logical memory into blocks of the same size called pages. When a program needs to access a memory page, the system checks a page table to find the corresponding frame in physical memory.

Segmentation, on the other hand, divides the program's logical memory into segments of different sizes, each of which has a particular logical meaning within the program (like a procedure or a data array). Each segment is then mapped into the system's physical memory whenever it is needed.

A hardware page-table length register (PTLR) is often used to check for validity of a logical address. The Page-Table Length Register (PTLR) is used in memory protection. It holds the size of the page table, which prevents a program from accessing memory outside its own page table. This is crucial for ensuring that one process cannot access the memory space of another, providing a key aspect of memory protection. The PTLR is checked every time a memory reference is made to ensure that the reference is to a valid page.

18B. Frame size of 256 bytes means an offset of 8 bits. The first 4 bits for the logical address **0100**11010010 are the page number, i.e., $0100_2 = 4$. Frame number for page 4 is 10011. Therefore, the physical address in the main memory is **10011**11010010, which is of 13 bits for a main memory of 8KB.

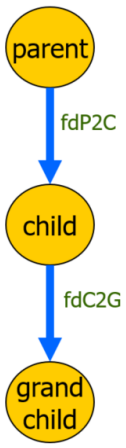
The logical address 010011010010 is 12 bits long. Given a frame size of 256 bytes, which is 2^8, the offset within a page (or frame) is 8 bits. Therefore, the remaining 4 bits of the logical address are used for the page number. We can split the logical address into page number and offset:

Page number: 0100 (which is 4 in decimal) Offset: 11010010. From the page table, we can see that page 4 maps to frame number 10011 (which is 19 in decimal). The physical address is then formed by concatenating the frame number with the offset. So, the physical address corresponding to the logical address 010011010010 is:

Frame number: 10011 Offset: 11010010. So, the physical address is 10011110010010.

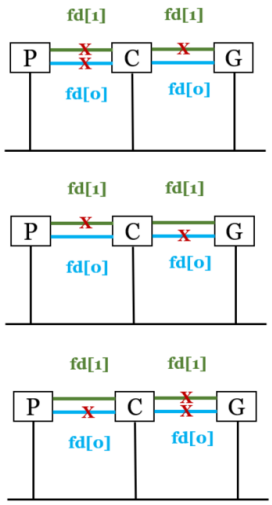
Section C (Question 19 & Question 20. (10 marks)

19.



- Program fragment with parent/child/grand child getting data from parent:

```
int fdP2C[2], fdC2G[2];
if (pipe(fdP2C) < 0) { printf("Pipe creation error\n"); exit(1); }
if (pipe(fdC2G) < 0) { printf("Pipe creation error\n"); exit(1); }
if (fork() == 0) { /* child */
    if (fork() == 0) { /* grand child */
        close(fdC2G[1]); close(fdP2C[0]); close(fdP2C[1]); /* close excessive */
        // grand child program here ...
        read read read
        close(fdC2G[0]); /* close grand child in */
    } else { /* child */
        close(fdP2C[1]); close(fdC2G[0]); /* close excessive */
        // child program here ...
        read write read write read write
        close(fdP2C[0]); close(fdC2G[1]); /* close child in and grand child out */
} else { /* parent */
    close(fdP2C[0]); close(fdC2G[0]); close(fdC2G[1]); /* close excessive */
    // parent program here ...
    write write write
    close(fdP2C[1]); /* close parent out */
}
```



20.

There are three processes and five possible outputs:
hello, hellohello, hellobye, hellohellobye, hellobyehello.