Lebanese University
Faculty of Sciences I

2nd session 2017/2018
Duration: 2 hours

I3303 / INFO324
Operating System II

Problem I — 18 points

In each of the following programs, it is assumed that the parent process has Pid = 100, and that there are no other active processes in the system than those created by the program. Give all possible display results obtained by running each of the following programs.

Program 1

```
void main(){
      fork();
      if(fork())
          printf("%d\n",fork());
}
```

(5)

Program 2

```
void main(){
      int x;
      x =  (fork() + fork()) * fork();
      printf("%d\n",x);
}
```

(8)

Program 3

```
void main(){
      int p[2], x = 0, y;
      if(fork()){
            pipe(p);
            x = getpid();
            write(p[1],&x,sizeof(int));
      }
      else{
            read(p[0],&y,sizeof(int));
            printf("%d %d",x,y);
      }
}
```

(5)

Problem II — 16 points

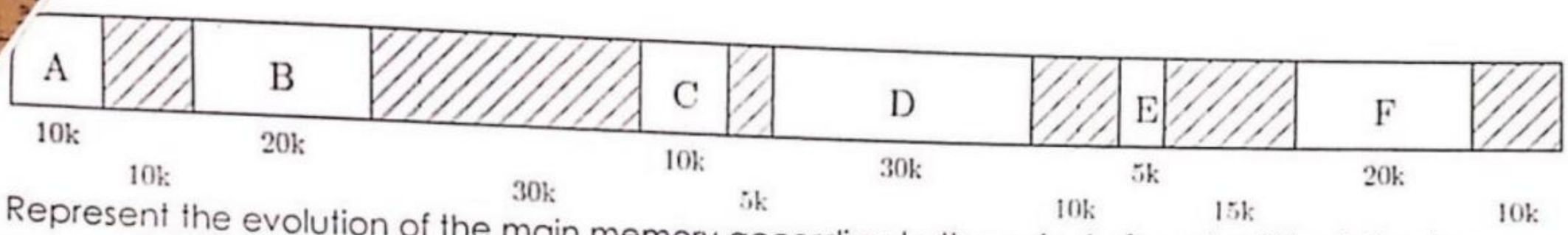
(6) 1. Using the SIGALRM signal, write a program that draws a number between 1 and 100 after one second and displays it.

(10) 2. Write a C program under UNIX where the parent process creates 100 child processes as follows: The father creates the first process and waits for the child to display its PID, before creating the second child. Then he waits for the second to display his PID before creating the third child, and so on.

Problem III — 16 points

1. We consider a memory managed by contiguous allocation. The allocation of processes is made according to the first fit algorithm. That is, the first encountered area whose size is greater than or equal to the size of the process to be loaded is the one that is allocated

the process. At the instant t, the state of the main memory is described in the figure below (where A, B, C, D, E, F are processes already loaded in memory, and the other parts are free zones):



Represent the evolution of the main memory according to the arrival of each of the following successive events.

- Event 1. Arrival of program G (20k).
- Event 2. Start of program B.
- Event 3. Arrival of program H (15k).
- Event 4. Departure of the program E.
- Event 5. Arrival of program I (40k).

2. The memory of a computer contains 4-page frames and, at the beginning, all the frames are empty. How many page faults produces the following page references 3, 4, 4,1, 5, 2, 3,1,4 using, respectively, FIFO, OPTIMAL, and LRU replacement algorithms? Justify your answers by showing the contents of the frames after each reference.

Problem IV

20 points

1. Describe (in pseudocode) the steps to be followed by the function **void Move (char * file_name, int f1, int f2)** which moves a file from a source directory (f1 is the index of its descriptor entry) to a destination directory (f2 is the index of its descriptor entry).

2. Describe (in pseudocode) the steps to be followed by the function **int Similar (int f, char * file_name_1, file_name_2)** which compares the contents of two files belonging to the opened directory with entry descriptor **f**. If both files have the same content, the function returns 1, and 0 if not.

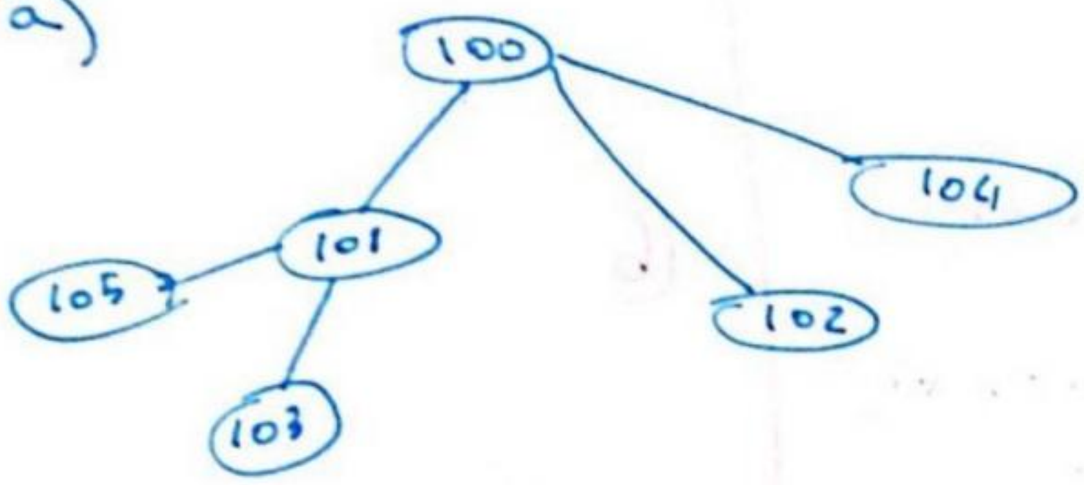
P.S: Feel free to use the structures and functions seen in the course.

Bon travail

I 3303 / 324 Second session 2018

Exercice 1

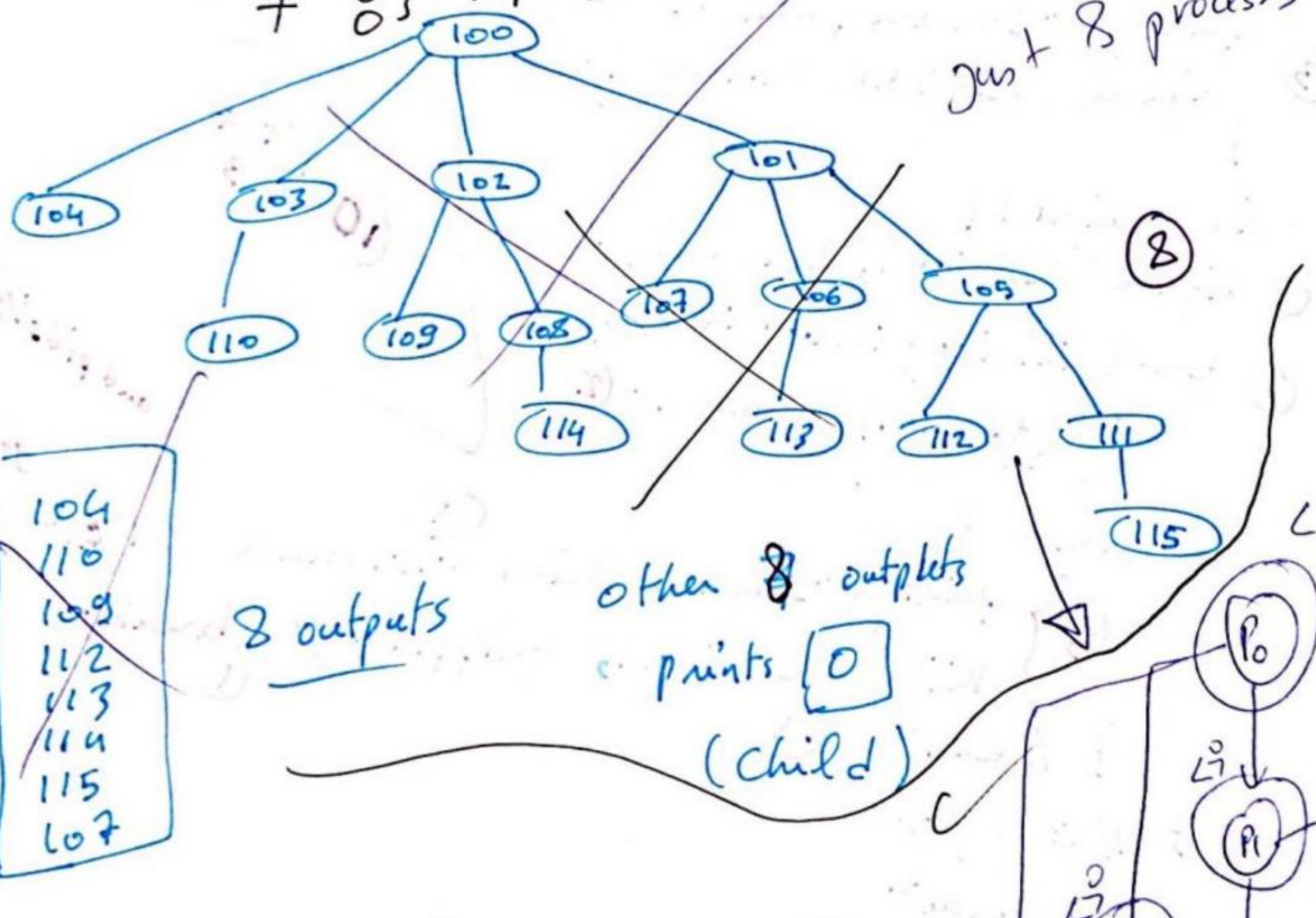
a)



(5)

output : 104
105
} pid of last 2 childs created (printed on father context)
+ 0 0 } → printed on child context

b)



just 8 processus

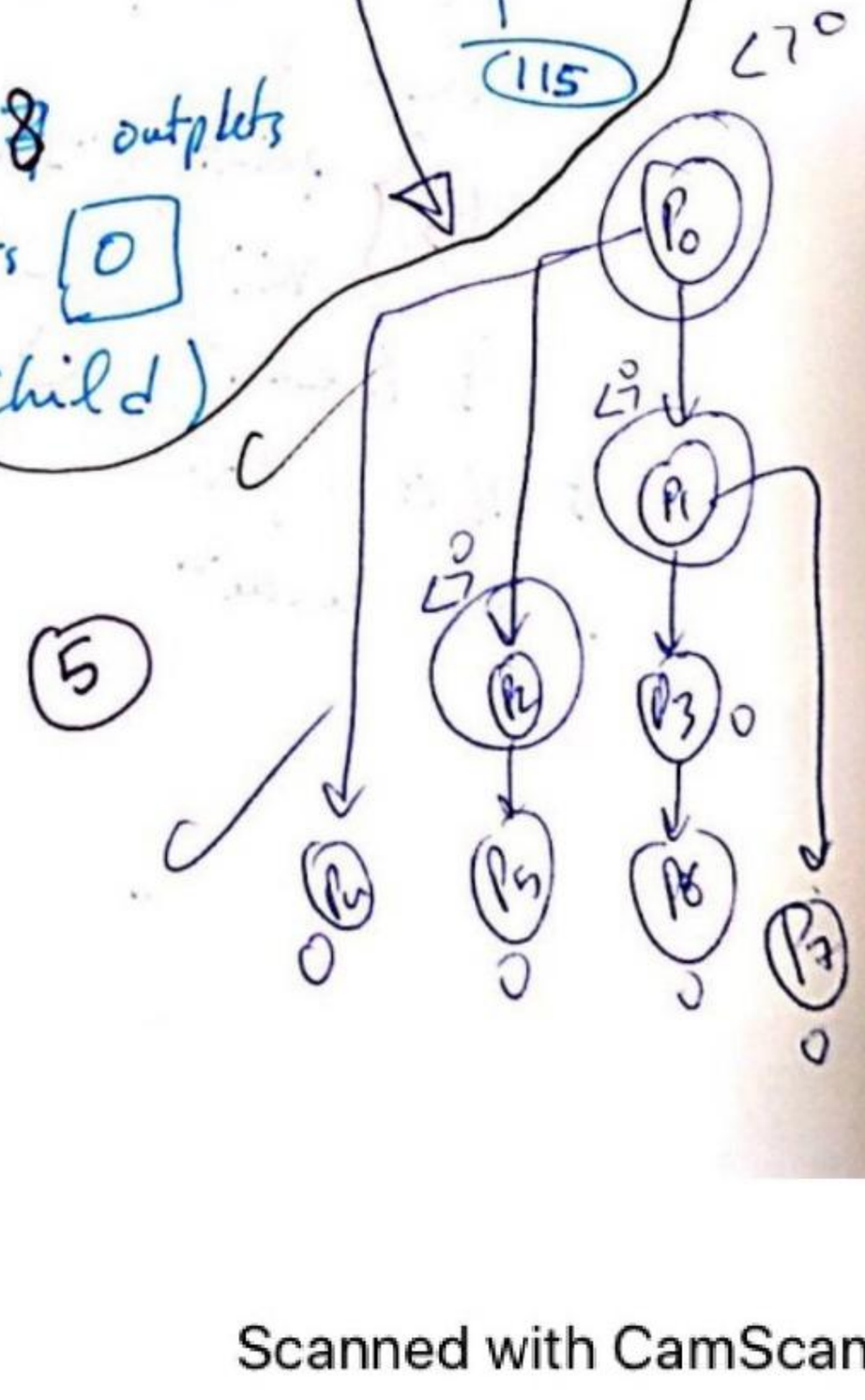
(8)

104
110
109
112
113
114
115
107

8 outputs

other 8 outputs
prints 0
(child)

c) the output is 0 0



(5)

(1)

Problem II :

a)

```
void handle(int nsig) { } (2)
void main() {
   srand (time(0));
(1) signal ( SIGALRM, handle);
(1) alarm(1);
(1) pause();
(1) int num = (rand() % 100) + 1;
(1) printf("%d\n", num);
}
```

(6)

b)

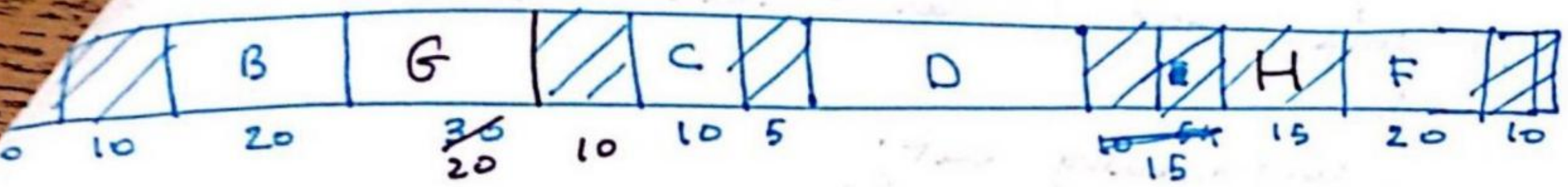
```
void handler (int nsig) {
(3)  printf ("I'm the parent and I got a message from my child %d\n",
      signal ( SIGUSR1, handler);
   }

void main() {
(1) signal ( SIGUSR1, handler);
(1) for (int i=0; i<100; i++) {
       if (fork()) pause(); (2)
       else {
(3)       printf("I'm the child with rank %d and pid: %d\n", i, getpid());
          kill ( SIGUSR1, getppid()); (1)
          break; (1)
       } // end if (1)
   } // End for
} // End main
```

(10) { 7 x 3

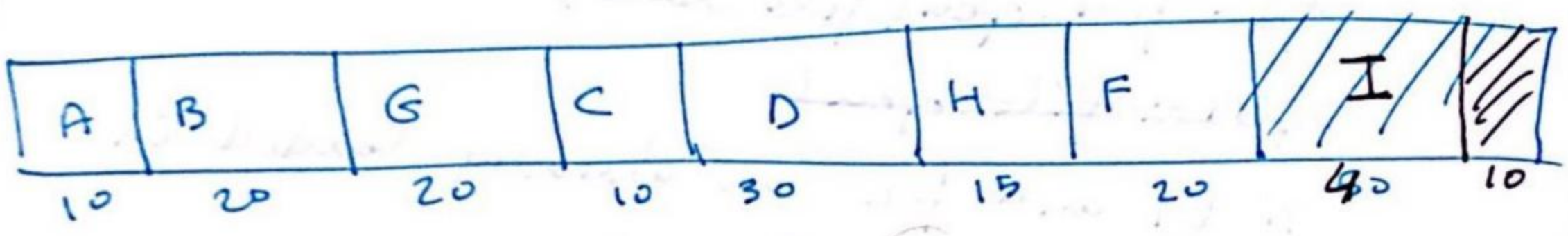
(1)

(16 pts) (5+11)



no place for Pg I (wok) ⇒ fragmentation (5)

⇒



2) $w = \{3, 4, 4, 1, 5, 2, 3, 1, 4\}$

FiFo (3)

ref	Fault	A	B	C	D
3	y	3			
4	y	3	4		
4	N	3	4		
1	y	3	4	1	
5	y	3	4	1	5
2	y	2	4	1	5
3	y	2	3	1	5
1	N	2	3	1	5
4	y	2	3	4	5

7 page faults

LRV (4)

ref	Fault				
3	y	3			
4	y	4	3		
4	N	4	3		
1	y	1	4	3	
5	y	5	1	4	3
2	y	2	5	1	4
3	y	3	2	5	1
1	N	1	3	2	5
4	y	4	1	3	2

7 page faults

optimal (3)

	3	4	4	1	5	2	3	1	4
	3	3	3	3	3	3	3	3	4
		4	4	4	4	2	2	2	2
				1	1	1	1	1	1
					5	5	5	5	5

(2)

6 page faults

optimal

Scanned with CamScanner

Pb II (20 pts)

1) void move.(char *file_name, int f1, int f

{

folder_entry ent;

inode binode[16]

int fd, inode-file;

(10)

fd = file-open(file_name)

~~fd = file-open~~

// f1 and f2 are already loaded into fdesc in memory

// Just we should save the link of inode of the file fr directory f1 to f2 in memory, then write to disk

// search in data of ftopo[f1] desc for the entry containing filename and inode-file and delete this entry

// fp->inode-file = fd.inode-nb;

// add new entry in the ftopo[f2] desc

write(f2, file_name);

write(f2, inode-nb);

}

similar (int f, char *file1, char *file2)

// f is the descriptor of directory loaded into memory

// fdes[f].topo contains the data block of f

int fd1 = open_file (file1);

int fd2 = open_file (file2);

(10)

char *c1, *c2;

~~if fd1.log <> fd2~~

if fdesc[fd1].log <> fdes[fd2].log

return 0;

else

{ while (~~and(fd1, c1)~~ or ! Eof(fd1))

{ read (fd1, c1);

read (fd2, c2);

~~if c1 != c2~~

if (! strcmp(c1, c2))

return 0;

}

return 1;

}

}

(3)