CL220/205 – Operating Systems Lab

Monday, July 06, 2020

Instructor

Aliza Saeed, Saba Naseem, Muhammad Usman Joyia, Mazhar Hussain, Awais Azam

Final Term Exam
Spring Semester 2020
Max Time:3 Hour and 20 minutes
Max Marks: 75
Exam Weight (Out of 100). 40-50

___18F-0326_____________ ________4A__________
Roll No Section

Instructions:

1. The paper consists of **Objective and Subjective part.**
2. You should submit only one PDF document**. Screenshots of solution should be embedded into this word file and convert it into PDF before submission**.
3. You must submit your solution before due time via Classroom. Submissions submitted after the due time shall not be considered.
4. If you don't finish every part of a question, don't worry! You can still submit what you've done to get marks based on your efforts.
5. In case of copied or plagiarized solutions in exam Or If a student provided help to another student during exam both will be awarded "F" grade and it will affect the student CGPA.
6. Viva of any student can be conducted by the instructor after conducting an online exam in case of any doubt.
7. This document should be submitted through LMS. But in worst case, you can email it within the deadline.
8. Name your file in format "f18-XXXX_Name".

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Total
Total Marks	20	10	5	5	5	15	15	**75**
Marks Obtained								

Question no. 2 Show all your working and paste high resolution pictures of solution

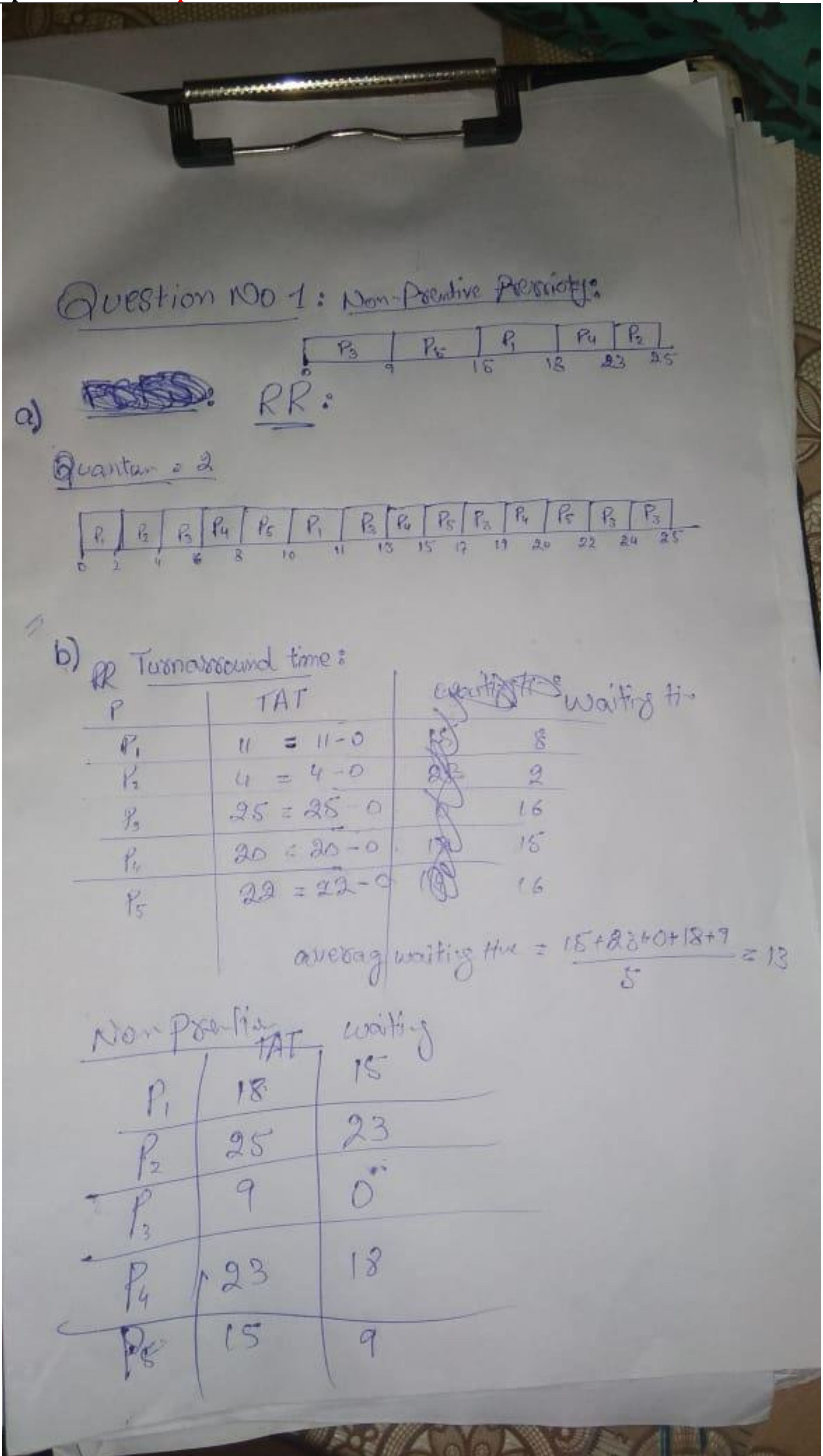
Note:
-Solution should be handwritten
-Do mention your Roll No (f18-XXXX) **on top of (every page) handwritten solution**
-Then do fill the answers in the given tables too. **(10)**

Consider the following set of processes, with the length of the CPU burst given in milliseconds:

Process	**Burst Time**	**Priority**
P1	3	2
P2	2	1
P3	9	4
P4	5	2
P5	6	3

The processes are assumed to have arrived in the order **P1, P2, P3, P4, P5**, all at time **0**.

a) Draw Gantt charts that illustrate the execution of these processes using the following scheduling algorithms: **Non-preemptive priority** (a larger priority number implies a higher priority), and **RR (quantum - 2)**



Question No 1: Non-Preentive Periority:

P_3	P_5	P_1	P_4	P_2
0	9	15	18	23	25

a) RR:

Quantum = 2

P_1	P_2	P_3	P_4	P_5	P_1	P_3	P_4	P_5	P_3	P_4	P_5	P_3	P_3
0	2	4	6	8	10	11	13	15	17	19	20	22	24	25

b) RR Turnarround time:

P	TAT	Waiting ti-
P_1	11 = 11 - 0	8
P_2	4 = 4 - 0	2
P_3	25 = 25 - 0	16
P_4	20 = 20 - 0	15
P_5	22 = 22 - 0	16

averag waiting time = $\frac{15+23+0+18+9}{5}$ = 13

Non Preentive	TAT	waiting
P_1	18	15
P_2	25	23
P_3	9	0
P_4	23	18
P_5	15	9

b) What is the turnaround time of each process for each of the scheduling algorithms in part **a**?

	Non-preemptive Priority	**RR**
P1	**18**	**11**
P2	**25**	**4**
P3	**9**	**25**
P4	**23**	**20**
P5	**15**	**22**

c) What is the waiting time of each process for each of these scheduling algorithms?

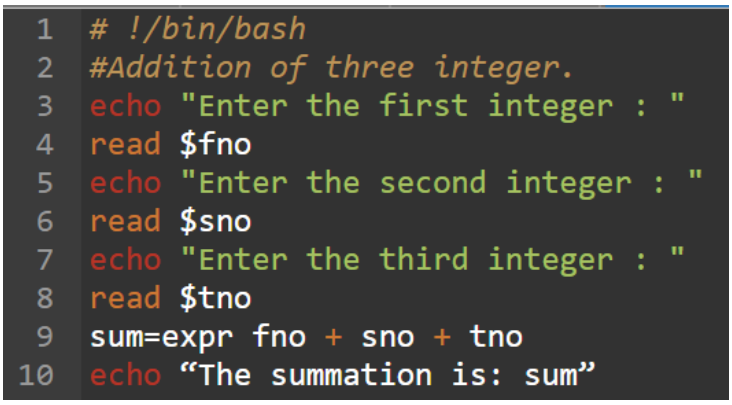
	Non- preemptive Priority	**RR**
P1	**15**	**8**
P2	**23**	**2**
P3	**0**	**16**
P4	**18**	**15**
P5	**9**	**16**

d) Which of the algorithms results in the minimum average waiting time (over all processes)?

	Non-preemptive Priority	RR
P1	**15**	**8**
P2	**23**	**2**
P3	**0**	**16**
P4	**18**	**25**
P5	**9**	**16**
Average WT	**13**	**13.4**

Question no. 3 **(5)**

a) Find the errors in the given shell script and fill the given table below.

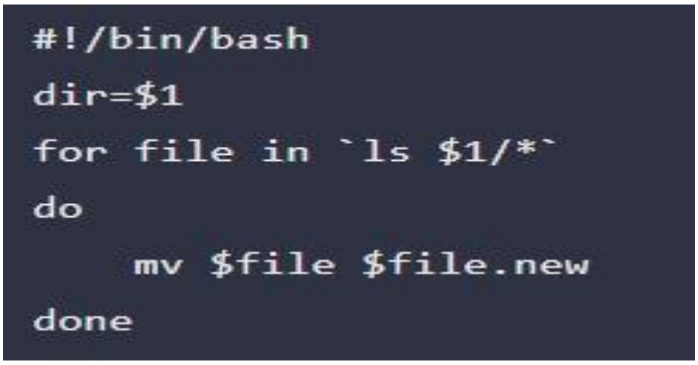


```
1  # !/bin/bash
2  #Addition of three integer.
3  echo "Enter the first integer : "
4  read $fno
5  echo "Enter the second integer : "
6  read $sno
7  echo "Enter the third integer : "
8  read $tno
9  sum=expr fno + sno + tno
10 echo “The summation is: sum”
```

Errors	Description with line number

Fno,sno,tno	In line 4,6,8 we don't use $for taking input in variables
Use of expr	In line 9 we have to use backtick for resolving expression an $ needed with variables
$ missing with sum	$ missing with echo to show output from variable

b) Explain the working of the code given below.



```
#!/bin/bash
dir=$1
for file in `ls $1/*`
do
      mv $file $file.new
done
```

It will rename all the files in the given directory with parameter $1

It will add .new at the end of each file in directory

Question no. 4 **(5)**

You have been hired as an official in a firm working on a project dealing with semaphores. As a junior worker you have been assigned to work on a module whose target output must be **XAYBZCXAYBZCXAYBZC…** This module comprises of three processes namely base process, middle process and higher end process where base process prints XA, middle process prints YB and higher end process prints ZC. You are allowed to use three semaphores. Your task is to set initial value of semaphores, sequence of processes and sequence of wait() and signal() calls in each process. You are not required to write complete code.

Code:

```
#include<stdio.h>
#include<pthread.h>
#include<stdlib.h>
#include<semaphore.h>

using namespace std;
sem_t s
sem_t  s2
sem_t  s3;
void *base(void *arg)
{
       sem_wait(&s);
       cout << "XA";
       sem_post(&s);
}
```

```
void *middle(void *arg)
{
	sem_wait(& s2);
	cout << "YB";
	sem_post(& s2);
}
void *higher(void *arg)
{
	sem_wait(& s3);
	cout << "ZC";
	sem_post(& s3);
}
int main()
{
	void *ptr = NULL;
	pthread_t th[3];
	sem_init(&s, 0, 1);
	sem_init(&s2, 0, 2);
	sem_init(&s3, 0, 3);
	for (int i = 0; i<3; i++)
	{
		pthread_create(&th[0], NULL, &base, NULL);
		pthread_join(th[0], &ptr);
		pthread_create(&th[1], NULL, &middle, NULL);
		pthread_join(th[1], &ptr);
		pthread_create(&th[2], NULL, &higher, NULL);
		pthread_join(th[2], &ptr);
	}

}
```

Question no. 5 (5)

Consider two processes A and B, each accessing two semaphores C and D, set to value 1. Sequence of accessing these two semaphores is given below.

A	B

wait(C)
wait(D)
.
.
.
signal(C)
signal(D)
	wait(D)
	wait(C)
	.
	.
	.
	signal(D)
	signal(C)

a) Show the sequence of wait and signal calls by A and B when the system enters the deadlock.

b) Show the sequence of wait and signal calls by A and B when the system does not enter any deadlock.

Question no. 6 Paste your code plus screenshots of output (15)

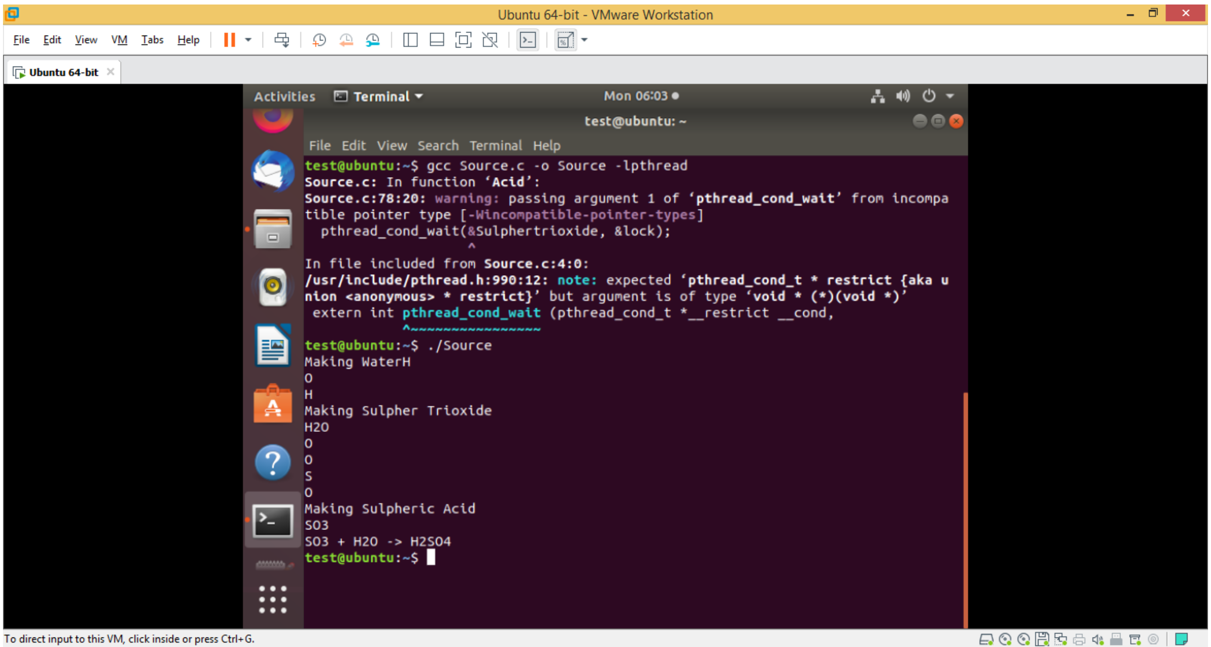
Code:

```
#include <stdio.h>
#include <stdlib.h>
```

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


pthread_mutex_t lock;
pthread_cond_t H;
pthread_cond_t O;
pthread_cond_t S;
pthread_cond_t water;
pthread_cond_t sulphertrioxide;
int watermol = 0;
int sulphermol = 0;
int acidmol = 0;
void *Hydrogen(void *var)
{
       pthread_mutex_lock(&lock);
       printf("H \n");
       pthread_cond_signal(&H);
       watermol++;
       pthread_mutex_unlock(&lock);
}
void *Oxygen(void *var)
{
       pthread_mutex_lock(&lock);
       printf("O\n");
       pthread_cond_signal(&O);
       watermol++;
       sulphermol++;
       pthread_mutex_unlock(&lock);
}
void *Sulpher(void *var)
{
       pthread_mutex_lock(&lock);
       printf("S\n");

       pthread_cond_signal(&S);
       sulphermol++;
       pthread_mutex_unlock(&lock);
}

void *Water(void *var)
{
       pthread_mutex_lock(&lock);
       while (watermol < 3)
{
       pthread_cond_wait(&H, &lock);
       pthread_cond_wait(&H, &lock);
       pthread_cond_wait(&O, &lock);
}
       printf("H2O\n");
       acidmol++;
       pthread_cond_signal(&water);
       pthread_mutex_unlock(&lock);
}
void *Sulphertrioxide(void *var)
{
       pthread_mutex_lock(&lock);
       while(sulphermol < 4)
{
       pthread_cond_wait(&O, &lock);
       pthread_cond_wait(&O, &lock);
       pthread_cond_wait(&O, &lock);
       pthread_cond_wait(&S, &lock);
}
```

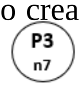
```
        printf("SO3\n");
acidmol++;
        pthread_cond_signal(&sulphertrioxide);
        pthread_mutex_unlock(&lock);
}
void *Acid(void *var)
{
        pthread_mutex_lock(&lock);
        while (acidmol < 2)
{
        pthread_cond_wait(&water, &lock);
        pthread_cond_wait(&Sulphertrioxide, &lock);
}
        printf("SO3 + H2O -> H2SO4\n");
        pthread_cond_signal(&sulphertrioxide);
        pthread_mutex_unlock(&lock);
}



int main()
{
        pthread_t th0;
        pthread_t th1;
        pthread_t th2;
        pthread_t th3;
        pthread_t th4;
        pthread_t th5;
        pthread_t th6;
        pthread_t th7;

        printf("Making Water");
        pthread_create(&th0, NULL, Hydrogen, NULL);
        pthread_create(&th1, NULL, Hydrogen, NULL);
        pthread_create(&th2, NULL, Oxygen, NULL);

        pthread_create(&th3, NULL, Water, NULL);

        pthread_join(th0, NULL);
        pthread_join(th1, NULL);
        pthread_join(th2, NULL);


        printf("Making Sulpher Trioxide\n");
        pthread_create(&th0, NULL, Sulpher, NULL);
        pthread_create(&th1, NULL, Oxygen, NULL);
        pthread_create(&th2, NULL, Oxygen, NULL);
        pthread_create(&th4, NULL, Oxygen, NULL);

        pthread_create(&th5, NULL, Sulphertrioxide, NULL);

        pthread_join(th0, NULL);
        pthread_join(th1, NULL);
        pthread_join(th2, NULL);
        pthread_join(th4, NULL);

        printf("Making Sulpheric Acid\n");

pthread_join(th3, NULL);
pthread_join(th5, NULL);
        pthread_create(&th6, NULL, Acid, NULL);
        pthread_join(th6, NULL);



        pthread_mutex_destroy(&lock);
}
```

Screenshot:


Ubuntu 64-bit - VMware Workstation
File Edit View VM Tabs Help
Ubuntu 64-bit
Activities
Terminal
Mon 06:03
test@ubuntu: ~
File Edit View Search Terminal Help
To direct input to this VM, click inside or press Ctrl+G.

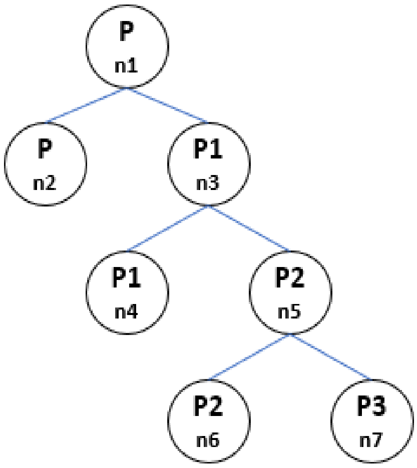
```
test@ubuntu:~$ gcc Source.c -o Source -lpthread
Source.c: In function ‘Acid’:
Source.c:78:20: warning: passing argument 1 of ‘pthread_cond_wait’ from incompa
tible pointer type [-Wincompatible-pointer-types]
  pthread_cond_wait(&Sulphertrioxide, &lock);
                    ^
In file included from Source.c:4:0:
/usr/include/pthread.h:990:12: note: expected ‘pthread_cond_t * restrict {aka u
nion <anonymous> * restrict}’ but argument is of type ‘void * (*)(void *)’
 extern int pthread_cond_wait (pthread_cond_t *__restrict __cond,
            ^~~~~~~~~~~~~~~~~
test@ubuntu:~$ ./Source
Making WaterH
O
H
Making Sulpher Trioxide
H2O
O
O
S
O
Making Sulpheric Acid
SO3
SO3 + H2O -> H2SO4
test@ubuntu:~$ 
```

Question no. 7 Paste your code plus screenshots of output (15)

a) Write a program to create the given process tree and print each node information, for example last node  in the tree that can be represented as **P3(n7)** where **P3** represents process name and **n7** represents the node name. You can print all nodes similarly as the last node sample output is as follow: => Process name: **P3** and Node name: **n7** *(6 marks)*

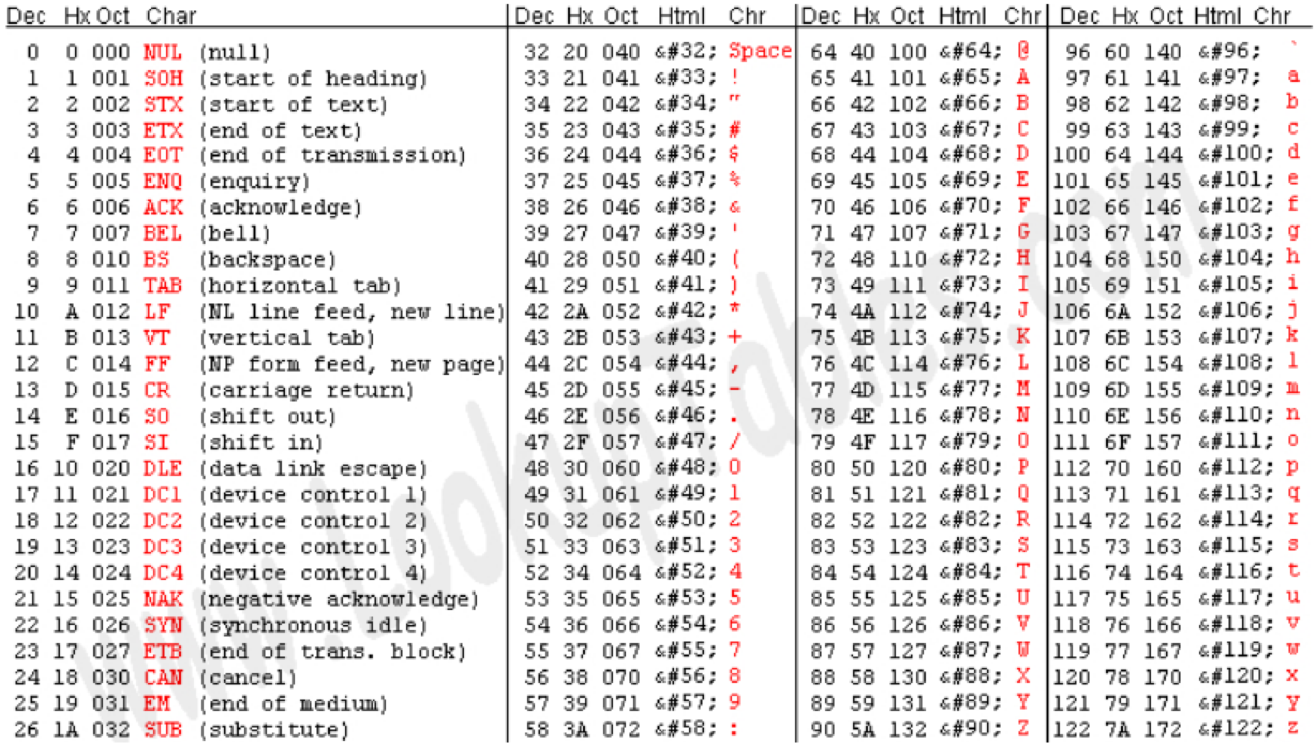
b) Process **P(n2)** communicate with process **P3(n7)**.

- **P(n2)** sends a message to **P3(n7)** using pipes mechanism of inter-process communication. *(2 marks)*
- **P3(n7)** converts the received message into **Invert Case** using **flipCase** function that will be called by **P3(n7)**. This flipCase function turns each lowercase character into an uppercase character and each uppercase character into a lowercase character. You are not allowed to use tolower() and toupper() functions. *(4 marks)*



For example, if the message string is “GNU Image Processing Tool-Kit8", then this function should return a string “gnu iMAGE pROCESSING tOOL-kIT8".

You will find it helpful to have reference of a table of ASCII codes for this problem as below:



Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040	&#32;	Space	64	40	100	&#64;	@	96	60	140	&#96;	`
1	1	001	SOH (start of heading)	33	21	041	&#33;	!	65	41	101	&#65;	A	97	61	141	&#97;	a
2	2	002	STX (start of text)	34	22	042	&#34;	"	66	42	102	&#66;	B	98	62	142	&#98;	b
3	3	003	ETX (end of text)	35	23	043	&#35;	#	67	43	103	&#67;	C	99	63	143	&#99;	c
4	4	004	EOT (end of transmission)	36	24	044	&#36;	$	68	44	104	&#68;	D	100	64	144	&#100;	d
5	5	005	ENQ (enquiry)	37	25	045	&#37;	%	69	45	105	&#69;	E	101	65	145	&#101;	e
6	6	006	ACK (acknowledge)	38	26	046	&#38;	&	70	46	106	&#70;	F	102	66	146	&#102;	f
7	7	007	BEL (bell)	39	27	047	&#39;	'	71	47	107	&#71;	G	103	67	147	&#103;	g
8	8	010	BS (backspace)	40	28	050	&#40;	(	72	48	110	&#72;	H	104	68	150	&#104;	h
9	9	011	TAB (horizontal tab)	41	29	051	&#41;	)	73	49	111	&#73;	I	105	69	151	&#105;	i
10	A	012	LF (NL line feed, new line)	42	2A	052	&#42;	*	74	4A	112	&#74;	J	106	6A	152	&#106;	j
11	B	013	VT (vertical tab)	43	2B	053	&#43;	+	75	4B	113	&#75;	K	107	6B	153	&#107;	k
12	C	014	FF (NP form feed, new page)	44	2C	054	&#44;	,	76	4C	114	&#76;	L	108	6C	154	&#108;	l
13	D	015	CR (carriage return)	45	2D	055	&#45;	-	77	4D	115	&#77;	M	109	6D	155	&#109;	m
14	E	016	SO (shift out)	46	2E	056	&#46;	.	78	4E	116	&#78;	N	110	6E	156	&#110;	n
15	F	017	SI (shift in)	47	2F	057	&#47;	/	79	4F	117	&#79;	O	111	6F	157	&#111;	o
16	10	020	DLE (data link escape)	48	30	060	&#48;	0	80	50	120	&#80;	P	112	70	160	&#112;	p
17	11	021	DC1 (device control 1)	49	31	061	&#49;	1	81	51	121	&#81;	Q	113	71	161	&#113;	q
18	12	022	DC2 (device control 2)	50	32	062	&#50;	2	82	52	122	&#82;	R	114	72	162	&#114;	r
19	13	023	DC3 (device control 3)	51	33	063	&#51;	3	83	53	123	&#83;	S	115	73	163	&#115;	s
20	14	024	DC4 (device control 4)	52	34	064	&#52;	4	84	54	124	&#84;	T	116	74	164	&#116;	t
21	15	025	NAK (negative acknowledge)	53	35	065	&#53;	5	85	55	125	&#85;	U	117	75	165	&#117;	u
22	16	026	SYN (synchronous idle)	54	36	066	&#54;	6	86	56	126	&#86;	V	118	76	166	&#118;	v
23	17	027	ETB (end of trans. block)	55	37	067	&#55;	7	87	57	127	&#87;	W	119	77	167	&#119;	w
24	18	030	CAN (cancel)	56	38	070	&#56;	8	88	58	130	&#88;	X	120	78	170	&#120;	x
25	19	031	EM (end of medium)	57	39	071	&#57;	9	89	59	131	&#89;	Y	121	79	171	&#121;	y
26	1A	032	SUB (substitute)	58	3A	072	&#58;	:	90	5A	132	&#90;	Z	122	7A	172	&#122;	z

c) Now **P3(n7)** communicate with process **P1(n4)** using pipes mechanism and sends the inverted case message to **P1(n4)** that display the received message. (3 marks)

Good Luck ☺