Program Code TU856
Module Code CMPU 2017

TECHNOLOGICAL UNIVERSITY DUBLIN

Grangegorman

BSc in Computer Science (International)

Year 2

SEMESTER 1 EXAMINATIONS 2022/23

Operating Systems 2

Internal Examiners

Mr. Denis Manley
Dr Paul Doyle

External Examiners

Ms. Sanita Tifenale

Answer Questions 1 and any two others

Question 1 is worth 40 marks, all the rest are worth 30.

1

a) Given the following arrival times and CPU time for 4 processes determine the average turnaround time for:

i. A Round Robin schedule algorithm with a time slice of 6 *ms*. **(4 marks)**

ii. The Shortest Remaining Time. **(4 marks)**

Arrival Time	0	1	2	3
Job	A	B	C	D
CUP cycle time	8	4	9	5

b) Explain, using a suitable example, how the little man computer (LMC) carries out an instruction cycle: the fetch and execute cycle. **(4 marks)**

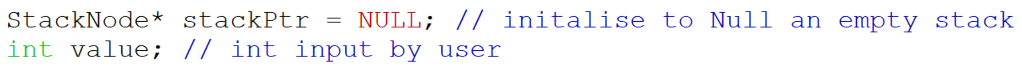
c) Explain, using a suitable example, the steps required to insert a node onto an empty queue.

(8 marks)

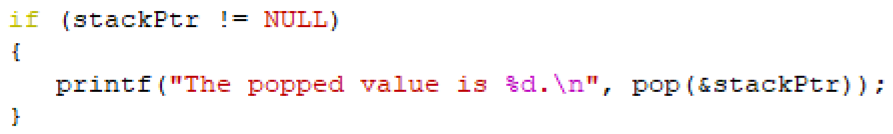
d) Explain, using suitable examples, the steps required to delete a node to an ordered linked list. **(10 marks)**

e) What will be the output the following code (delete function) if the stack has 2 nodes and this delete function is called twice? Clearly, explain your reasoning **(10 marks)**

The function call and the parameters passed to the delete function are:



```
StackNode* stackPtr = NULL; // initalise to Null an empty stack
int value; // int input by user
```



```
if (stackPtr != NULL)
{
   printf("The popped value is %d.\n", pop(&stackPtr));
}
```

The code for the delete function is:

```
// remove a node from the stack top
int delete(StackNode* *topPtr)
{


   StackNode* tempPtr = *topPtr;

   int popValue = (*topPtr)->data;

   topPtr = (*topPtr)->nextPtr;

   free(tempPtr);


  return popValue;
}
```

2

a) Distinguish between single and multi-threading processes. **(4 marks)**

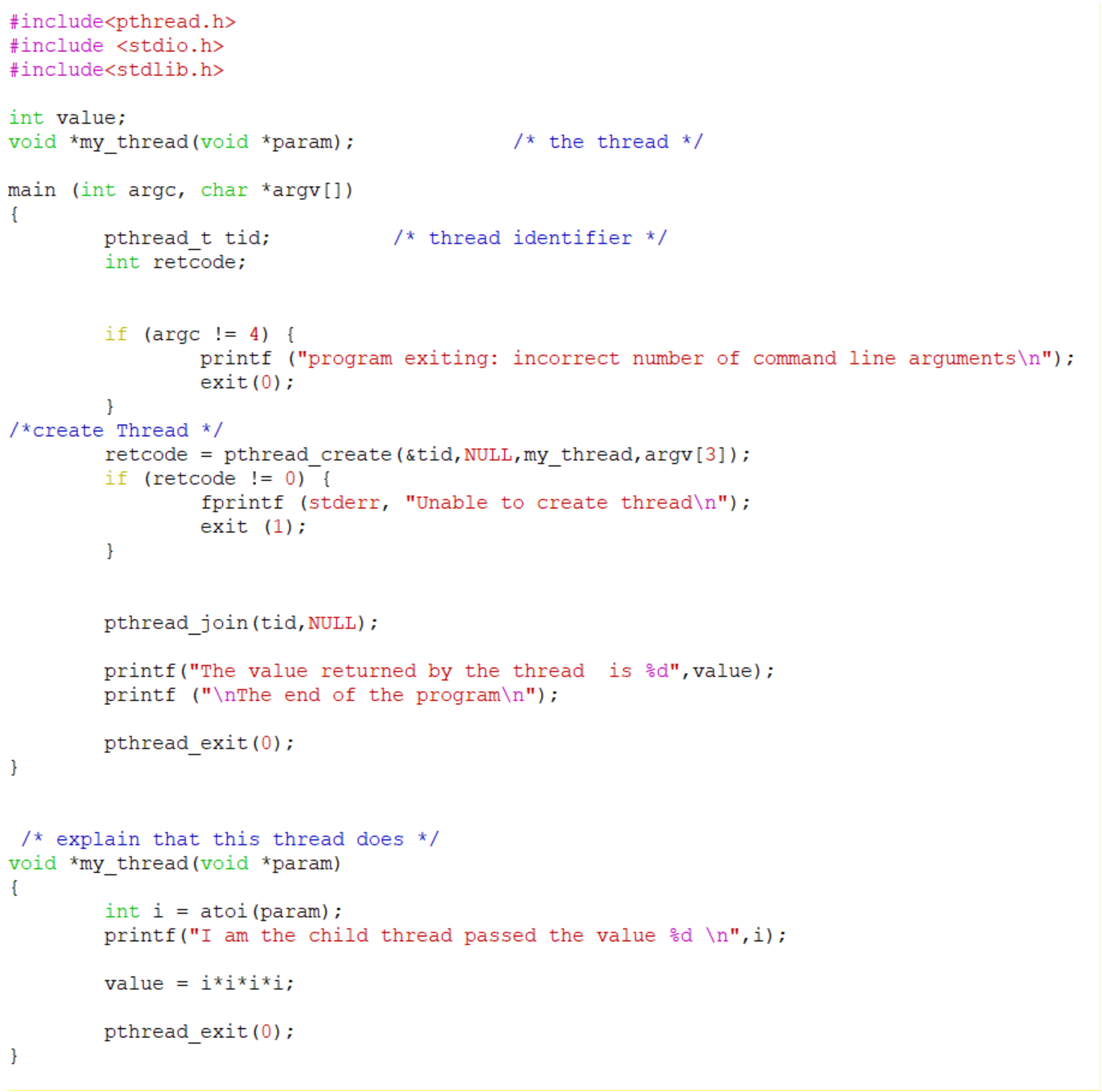
b) In C, a thread is created using the following code:

*int pthread_create(pthread_t *tidp, pthread_attr_t *attr, *start_rtn, void * arg)*

Clearly explain what each of the arguments in the thread create function mean.

(8 marks)

c) Explain, in your own words, the following code: **(10 marks)**



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

int value;
void *my_thread(void *param);              /* the thread */

main (int argc, char *argv[])
{
        pthread_t tid;          /* thread identifier */
        int retcode;


        if (argc != 4) {
                printf ("program exiting: incorrect number of command line arguments\n");
                exit(0);
        }
/*create Thread */
        retcode = pthread_create(&tid,NULL,my_thread,argv[3]);
        if (retcode != 0) {
                fprintf (stderr, "Unable to create thread\n");
                exit (1);
        }


        pthread_join(tid,NULL);

        printf("The value returned by the thread  is %d",value);
        printf ("\nThe end of the program\n");

        pthread_exit(0);
}


 /* explain that this thread does */
void *my_thread(void *param)
{
        int i = atoi(param);
        printf("I am the child thread passed the value %d \n",i);

        value = i*i*i*i;

        pthread_exit(0);
}
```

d) What would be the output of the code in part c if the following are input at the command prompt? **(6 marks)**

(a.) ./thread
(b.) ./thread 3 2 4
(c.) ./thread 3 5 7

e) What would be the two outcomes in the above program if the *pthread_join* command was removed and the command line input was *./thread 6 5 7*? Explain the reason for your answer. **(2 marks)**

3:

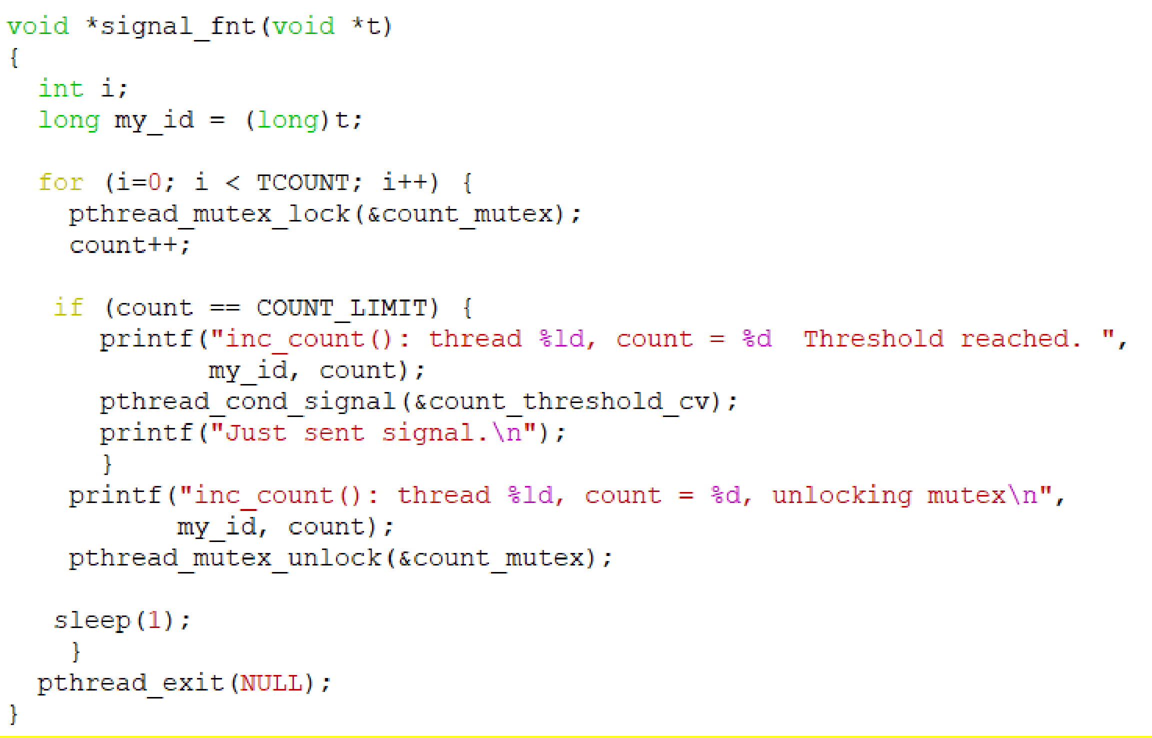
a) Explain, using an example, why it is critical to ensure that concurrency is carefully controlled for processes accessing the same data item; in other words the *race* problem.

(6 marks)

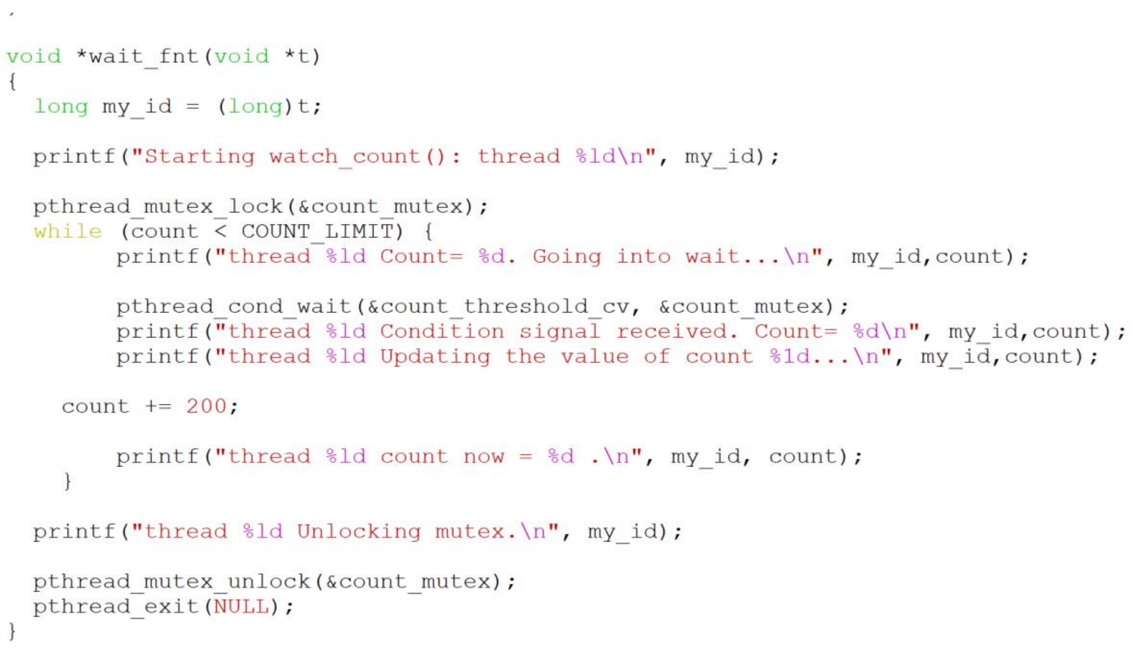
b) Two ways to prevent the race problem are Test and Set and Wait and Signal. Distinguish between each approach. **(4 marks)**

c) Explain, in detail, what the following two threads are doing: **(12 Marks)**

assume t-count =4, count = 0 and COUNT_LIMIT = 6

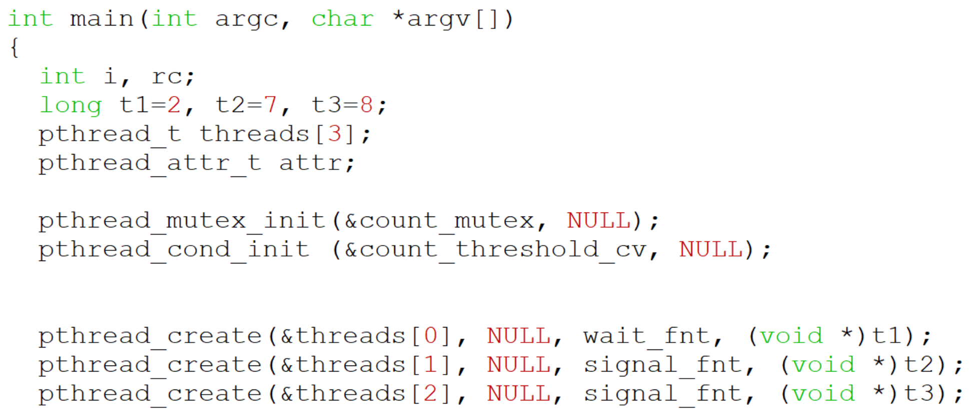


```
void *signal_fnt(void *t)
{
  int i;
  long my_id = (long)t;

  for (i=0; i < TCOUNT; i++) {
    pthread_mutex_lock(&count_mutex);
    count++;

   if (count == COUNT_LIMIT) {
      printf("inc_count(): thread %ld, count = %d  Threshold reached. ",
             my_id, count);
      pthread_cond_signal(&count_threshold_cv);
      printf("Just sent signal.\n");
      }
    printf("inc_count(): thread %ld, count = %d, unlocking mutex\n",
           my_id, count);
    pthread_mutex_unlock(&count_mutex);

   sleep(1);
    }
  pthread_exit(NULL);
}
```



```
void *wait_fnt(void *t)
{
  long my_id = (long)t;

  printf("Starting watch_count(): thread %ld\n", my_id);

  pthread_mutex_lock(&count_mutex);
  while (count < COUNT_LIMIT) {
        printf("thread %ld Count= %d. Going into wait...\n", my_id,count);

        pthread_cond_wait(&count_threshold_cv, &count_mutex);
        printf("thread %ld Condition signal received. Count= %d\n", my_id,count);
        printf("thread %ld Updating the value of count %1d...\n", my_id,count);

    count += 200;

        printf("thread %ld count now = %d .\n", my_id, count);
    }

  printf("thread %ld Unlocking mutex.\n", my_id);

  pthread_mutex_unlock(&count_mutex);
  pthread_exit(NULL);
}
```

d) If *main()* has the following thread create calls:



```
int main(int argc, char *argv[])
{
  int i, rc;
  long t1=2, t2=7, t3=8;
  pthread_t threads[3];
  pthread_attr_t attr;

  pthread_mutex_init(&count_mutex, NULL);
  pthread_cond_init (&count_threshold_cv, NULL);


  pthread_create(&threads[0], NULL, wait_fnt, (void *)t1);
  pthread_create(&threads[1], NULL, signal_fnt, (void *)t2);
  pthread_create(&threads[2], NULL, signal_fnt, (void *)t3);
```

Give a sample output of this multithread program and explain your reasoning.

(8 marks)

4

a) Identify the *four* conditions necessary for *Deadlock* to occur. **(4 Marks)**

b) Deadlock detection / recovery is another way of handling deadlock; explain, using an example the steps involved to reduce a deadlock detection directed resource graphs..
(**8 Marks)**

c) A possible consequence of deadlock prevention is starvation. What is starvation and how operating systems prevent starvation? **(4 marks)**

d) A second method of dealing with deadlock is deadlock prevention. Using deadlock prevention Bankers algorithm answer the following:

	Allocation	Max	Need	Available
				3 2 2 1
	A B C D	A B C D	A B C D	A B C D
P0	4 0 0 1	7 0 2 1		
P1	1 1 0 0	1 6 5 0		
P2	1 0 4 5	3 3 4 6		
P3	0 4 2 1	1 5 6 2		
P4	0 3 1 2	2 4 3 2		

i. How many resources of type A, B, C and D are there? **(2 marks)**

ii. What are the contents of the *Need* column? **(2 marks)**

iii. Is the system in a safe state? Provide reasoning for your answer.
(5 marks)

iv. If a request from process P2 arrives for additional resources of {0, 2, 0, 0}, will the system be in a safe state? Provide reasoning for your answer. **(5 marks)**