


National University of Computer and Emerging Sciences, Lahore Campus

	Course:	Operating Systems	Course Code:	CS-220/205
	Program:	BS(Computer Sciences)	Semester:	Spring 2020
	Duration:	180+15 minutes	Total Marks:	50
	Paper Date:	22 nd June, 2020	Weight:	50(4A,4B)% 45(4C,4D,6A)%
	Section:	ALL	Page(s):	3
	Exam:	Final	Roll No.	

Instructions/Notes:

- Read all the questions carefully, all questions are compulsory.
- Submit the solution at Learning Management System (LMS). Here LMS is Google Classroom, and it is the only preferred submission mechanism.
- Submission should be in the form of a SINGLE PDF file.
- Above mentioned 15 minutes are included only for the submission. The exam time is 180 minutes.
- Write your answers on a paper clearly so the scanned copies are clear and legible (readable).
- Readability is the responsibility of the student, if an answer is not readable, then it will be marked zero and the responsibility is upon the student.
- While creating the PDF put all the images in order, meaning answer of one question should be on contiguous pages.
- Write the page number with your own hand on each page of your answer book.
- No rough work should be included in the answer book. All pages uploaded to the LMS will be considered a valid answer.
- If your unintelligible rough work appears in the answer book along with the real answer, then the question will be marked zero.
- Duplicate answers to the same question will be marked zero.
- Whenever a numerical is under question you have to elaborate the steps taken to calculate the answer to the question.
- Resubmission is allowed, but only the last submission will be counted. Previous submissions will be discarded and will not be considered.
- **Reading the above guidelines is mandatory, not reading does not exonerate a student of any repercussions.**

Question 1 (10 points): You have to implement a function (`merge_four`) which merges four sorted arrays into one sorted array. You are already given a function that merges two sorted arrays into one (`merge_two`). Use that function (`merge_two`) to merge four arrays into one. Use threads to complete the task in parallel. Assume that the length of all four input arrays is equal. There are no duplicates in any array.

NOTE: 70% marks go to how you create and manage the threads and allow maximum parallelism. 20% marks go to how you manage the memory. 10% marks go to elegance of the solution.

```
#include<stdio.h>
//Other includes ... needless to mention
struct params{
    int* arr_1;
    int* arr_2;
    int* result;
    int size;
};

void* merge_two(void * args)
{
    int* arr_1 = ((struct params)args).arr_1;
    int* arr_2 = ((struct params)args).arr_2;
    int* result = ((struct params)args).result;
    const int N = ((struct params)args).size;
    for (int i = 0, j = 0, k = 0; i < N && j < N; )
    {
        if(arr_1[i] > arr_2[j])
        {
            result[k] = arr_2[j];
            k++;
            j++;
        }
        else
        {
            result[k] = arr_1[i];
            k++;
            i++;
        }
    }
    while ( i < N)
        result[k++] = arr_1[i++];

    while ( j < N)
        result[k++] = arr_2[j++];

    return NULL;
}

void merge_four (int* arr_1, int* arr_2, int* arr_3, int* arr_4, int size)
{
    int* res_1 = new int[2*size];
    int* res_2 = new int[2*size];
    int* result = new int[4*size];
    // Write the complete code of merger_four() function on your answer book, including
    // the lines written here in the question.
}
```

Question 2 (5 points): Consider multiple threads executing the following two functions. These threads print a string containing any number of a's and b's in an order. Synchronize the threads (using Semaphores) so that the string becomes a concatenation of the substring "ab". You are not allowed to use anything but semaphores and if conditions. Do not forget to mention the initial values of semaphores and variables. Following are few examples:

- ab (correct)
- ab ab (correct)
- ba (incorrect)
- ab ba (incorrect)

Function 1		Function 2	
1	<code>void function_1()</code>	1	<code>void function_2()</code>
2	<code>{</code>	2	<code>{</code>
3	<code> cout << "a";</code>	3	<code> cout << "b";</code>
4	<code>}</code>	4	<code>}</code>

Question 3 (10 points): Now change the same functions given above so that the string becomes a concatenation of the substring "abb". Following are few examples:

- abb (correct)
- abb abb (correct)
- ab (incorrect)
- ab ab (incorrect)
- bba (incorrect)
- bba bba (incorrect)

Question 4 (10 points): Suppose the size of a file is equal to $\langle \text{your} - \text{registration} - \text{year} \rangle \times \langle \text{your} - \text{registration} - \text{number} \rangle$ bytes. For example, a students registration year is 2014 and the registration number allocated to them is 4111. Then the size of the file is $14 \times 4111 = 57,554$ bytes. You are required to use multilevel indexing scheme with a block size of 256 bytes in the Index allocation method. Assuming a pointer size to be 4 bytes, then find out the following:

1. Minimum number of levels of multilevel indexing required to map this file, i.e. 1,2,3,4,5 etc. level of indexing.
2. Number of Index blocks to map this file.
3. Number of data blocks to map this file.
4. Show your working with the help of a diagram.

Question 5 (10 points): If you have a page table given below, then compute the physical address for the following parameters

- logical address: 45
- page-size: 8 words

0	10
1	0
2	9
3	1
4	8
5	2
6	7
7	3
8	6

Question 6 (5 points): If memory access time is equal to the year you enrolled in FAST, e.g. 18 nanoseconds, and page fault overhead time is equal to your registration number, e.g. 4111 nanoseconds. If the probability of having a page fault is 0.85, then calculate the Effective Access Time. Here, the page fault overhead time also includes the page swap in and swap out time.