BAHRIA UNIVERSITY (KARACHI CAMPUS)

OPEN ENDED LAB II – **Fall22**

(System Programing (LAB) **CSC-454)**

Class: BSE [4]-5 (B) (Morning)

Course Instructor: **Engr Rizwan Fazal / Engr Rehan Baig** Time Allowed: **1.5 Hour**

Max Marks: **6**

Student's Name: Ameer Hamza Enrollment : 02-131202-037

Instructions:

1. Submit your answers within file against each question with screenshot of both code and solution output.
2. File must be submitted in .pdf.

[CLO#05, 6 marks]

SCENARIO:

You are working as a system engineer in a Microsoft vendor company that creates Apps for Microsoft store.
Your Project manager assigned you a task to design an application for code editor for Microsoft store. For that you need to analyze the basics of NotePad/WordPad applications that comes built-in with Microsoft windows. You need to create a process and analyze the following for notepad and WordPad.

Q1: Run a loop or Use Recursion which enable program to print 5 times following for both Notepad and WordPad **(versionId, ThreadId, processId)**, meanwhile use exit thread function that-should be interrupt when counter reaches on 4rth iteration. (**4 Marks**)

Solution:

```
#include <iostream>

#include <iostream>

#include <thread>

#include <unistd.h>



void print_info(int iteration) {

   if (iteration == 4) {

      exit(0);

   }

   std::string app_name;

   if(iteration % 2 == 0)

      app_name = "Notepad";

   else

      app_name = "WordPad";

   std::cout << "Name: " << app_name << ",Thread ID: " << std::this_thread::get_id() << ",Process ID: "
<< getpid() << std::endl;

}



int main() {

   for (int id = 0; id <= 4; id++) {
```

```
        std::thread t(print_info, id);

        t.join();

    }

    return 0;
```

}**Output:**

```
Name: Notepad,Thread ID: 23236910225152,Process ID: 32481
Name: WordPad,Thread ID: 23236910225152,Process ID: 32481
Name: Notepad,Thread ID: 23236910225152,Process ID: 32481
Name: WordPad,Thread ID: 23236910225152,Process ID: 32481
```

Q2: Write a code for any two synchronization objects from following. (**2 Marks**)

1. Events
2. Semaphores
3. Mutexes

Solution:

```
#include <iostream>
#include <thread>
#include <mutex>
#include <semaphore.h>

std::mutex mtx;
sem_t sem;
int buffer[10];
int counter = 0;

void producer() {
    for (int i = 0; i < 5; i++) {
        sem_wait(&sem);
        mtx.lock();
        buffer[counter++] = i;
        std::cout << "Produced: " << i << std::endl;
        mtx.unlock();
        sem_post(&sem);
```

```
    }
}

void consumer() {
    for (int i = 0; i < 5; i++) {
        sem_wait(&sem);
        mtx.lock();
        std::cout << "Consumed: " << buffer[--counter] << std::endl;
        mtx.unlock();
        sem_post(&sem);
    }
}

int main() {
    sem_init(&sem, 0, 1);

    std::thread t1(producer);
    std::thread t2(consumer);

    t1.join();
    t2.join();

    sem_destroy(&sem);
    return 0;
}
```

Output:

```
Consumed: 0
Consumed: 0
Consumed: 0
Consumed: 0
Consumed: 0
Produced: 0
Produced: 1
Produced: 2
Produced: 3
Produced: 4
```