CG2271 Real Time Operating Systems
Lab 3 - Real Time Software Architectures
Answer Book

Name: Deepak Buddha	Matric Number: A0155454A
Name: Anton Chua	Matric Number: A0161811N

Submission Deadline: Friday 13 Oct 2359
Submit under IVLE → Files → Lab3-Submissions

Question 1 (1 marks)

The output is
9
8
7
6
5
4
3
2
1
0

The numbers (0-9) are displayed in descending order from top to bottom.

Question 2 (3 marks)

The numbers appear in the reverse order that they were enqueued. This is because the priority number for each item enqueued was actually decreasing (e.g. item '0' had priority no.9 and item '9' had priority no.0. According to prior.h file, the priority of the item is higher if it has a lower priority no. , so priority no.0 is the highest priority and priority no. 9 is the lowest. Hence when the items are dequeued, the highest priority item (item '9' in this example) is dequeued first until the item with lowest priority(item '0') , resulting in reverse order of which each item was enqueued.

Question 3 (2 marks)

In C++, void * is a void pointer that is a generic pointer that can allow a reference to any variable with any datatype.

It is needed in the enqueue function as the queue can only type void* references of items and by doing (void *)i, it is sending the reference of i as a type void * into the queue.

Question 4 (2 marks)

When the item is dequeued, it is dequeued as datatype void* and if we want to dereference the item through the void pointer to a specific variable (int val in the example), we must explicitly typecast it to the datatype required.

Question 5 (12 marks)

```
#include <Arduino.h>
#include "prioq.h"

#define PIN_LED6 6
#define PIN_LED7 7
```

```
boolean FLAG_int0 = false;
boolean FLAG_int1 = false;
unsigned long last_interrupt_time0 = 0;
unsigned long last_interrupt_time1 = 0;


// Declares a new type called "funcptr"

typedef void (*funcptr)(void);

TPrioQueue *queue;


// Flashes LED at pin 6: 5 times at 5 Hz
void int0task()
{
      for (int i=0; i<5; i++) {
            analogWrite(PIN_LED6, 255);
            delay(100);
            analogWrite(PIN_LED6, 0);
            delay(100);
            }
}

// Flashes LED at pin 7: 5 times at 1HZ
void int1task()
{
      for (int i=0; i<5; i++) {
            analogWrite(PIN_LED7, 255);
            delay(500);
            analogWrite(PIN_LED7, 0);
            delay(500);
            }
}

void int0ISR()
{
      funcptr task0 = int0task;
       unsigned long interrupt_time0 = millis();
       if (interrupt_time0 - last_interrupt_time0 > 200) { //
de-bouncing
          enq(queue, (void *)task0, 1); // enqueue task, higher
number lower priority
          if (interrupt_time0 > 200) {
             last_interrupt_time0 = interrupt_time0;
           }
```

```
      }
}

void int1ISR()
{
      funcptr task1 = int1task;
      unsigned long interrupt_time1 = millis();
       if (interrupt_time1 - last_interrupt_time1 > 200) { //
de-bouncing
          enq(queue, (void *)task1, 0); //enqueue task, lower
number higher priority
          if (interrupt_time1 > 200) {
             last_interrupt_time1 = interrupt_time1;
           }
       }
}

void setup()
{
      pinMode(PIN_LED6, OUTPUT);
      pinMode(PIN_LED7, OUTPUT);
      attachInterrupt(0, int0ISR, RISING);
      attachInterrupt(1, int1ISR, RISING);
      queue=makeQueue(); // set up queue
}

// Dequeues and calls functions if the queue is not empty
void loop()
{
      if(qlen(queue)>0) {
            funcptr doTask;
            doTask = (funcptr)deq(queue);
            doTask();
      }
}
```

Question 6 (2 marks)

By using the 'funcptr' function pointer.
It is declared as typedef void (*funcptr)(void); this means that we have a function pointer called funcptr for void input argument functions that return void as well. By using typedef, we can also use funcptr directly as a datatype.

Then we can pass the reference of functions into the priority queue as well by
1. Assigning the desired function to enqueue to functor : funcptr task1 = int1task;
2. Casting that funcptr variable as (void *) in the enq method
: enq (queue_name, (void *)task1, priority_number);

This sends the reference of the function as a void * type.

To dequeue the function
1. deq(queue) , this dequeues the reference of the function with the highest priority but the reference is of (void *) type
2. We create a functptr variable to hold the dequeued function and we typecast the dequeued function with funtctpr as it is void * type .
   eg : funcptr doTask = (funcptr)deq(queue_name);

This will allow us to dequeue the function reference as a function pointer and we can invoke it through the funcptr variable , eg. doTask();

Question 7 (2 marks)

Pin 7 LED flashes 5 times first and then Pin 6 LED flashes 5 times. This is expected as INT1 task has higher priority than INT0 task, so when button connected to INT1 is pressed , ISR1 is triggered and this in turn enqueues reference of INT1task into the queue with higher priority (no. 0) and when the button connected to INT0 is pressed, ISR0 is triggered and this enqueues reference of INT0 into the queue with lower priority (no.1). So when the functions are dequeued, INT1 is dequeued first and is executed. Only after INT1 has executed fully, INT0 is dequeued and then is executed.

Question 8 (2 marks)

Pin 6 LED flashes first and then Pin 7 LED. This is because when button connected to INT0 is pressed, ISR0 is triggered and enqueues reference of INT0 into the queue with priority no.0 but since INT1 button has not been pressed, the reference of INT0 is the only item in the queue and is dequeued first and INT0 task is executed. While INT0 task is executed, INT1 button is pressed and triggers ISR1 to enqueue reference of INT1 and is again the only item in the queue and is dequeued after INT1 task has finished executing.

Question 9 (4 marks)

No Pin 6 LED never flashes before Pin 7 LED. It is observed that upon pressing as suggested in the question, Pin 7 LED flashes 25 times first and then Pin 6 LED flashes 25 times. This is because reference of INT1 function is always enqueued into the queue with a higher priority than the reference of INT0 function.

When the INT1 button is pressed the first time, ISR1 is triggered and INT1 function reference is enqueued with higher priority (no.0) and is immediately dequeued, resulting in execution of INT1. As INT1 is executing, INT0 button is registered and triggers ISR0 which enqueues INT0 with lower priority(no.1). However while INT1 is still executing from the first button press, the second INT1 button press is registered causing INT1 function to be enqueued with higher priority(no.0) again.

So now the queue has INT1 (high priority) and INT0 (low priority) function references which will lead to INT1 being dequeued again over INT0 and this will result INT1 task being executed again. This sequence of events is propagated in the same way for all 5 button presses of INT1 and INT0, each round resulting an INT0 to be queued but never dequeued until all the INT1 has been executed.

Once all 5 INT1 tasks have been executed, the queue only contains INT0 function references and they are dequeued resulting in INT0 task being executed 5 times.