```
/*assignment 2- GPIO driver application*/

/*
 * labgpio.hpp
 *
 *  Created on: 14-Feb-2018
 *      Author: Kaustubh
 */
#include <stdint.h>
#include "LPC17xx.h"

#ifndef LABGPIO_0_HPP_
#define LABGPIO_0_HPP_

class LabGPIO_0
{
   private:
        uint8_t pinNum=0;
        uint8_t portNum=0;
   public:
      /**
       * You should not modify any hardware registers at this point
       * You should store the port and pin using the constructor.
       */

      /* @param {uint8_t} pin  - pin number between 0 and 32
       */

      LabGPIO_0(uint8_t,uint8_t);
      /**
       * Should alter the hardware registers to set the pin as an input
       */
      void setAsInput(void);
      /**
       * Should alter the hardware registers to set the pin as an output
       */
      void setAsOutput(void);
      /**
       * Should alter the set the direction output or input depending on the input.
       *
       * @param {bool} output - true => output, false => set pin to input
       */
      void setDirection(bool output);
      /**
       * Should alter the hardware registers to set the pin as high
       */
      void setHigh(void);
      /**
       * Should alter the hardware registers to set the pin as low
       */
      void setLow(void);
      /**
       * Should alter the hardware registers to set the pin as low
       *
       * @param {bool} high - true => pin high, false => pin low
       */
      void set(bool high);
```

```
    /**
     * Should return the state of the pin (input or output, doesn't matter)
     *
     * @return {bool} level of pin high => true, low => false
     */
    bool getLevel(void);
    ~LabGPIO_0();
};


#endif /* LABGPIO_0_HPP_ */

/*
 * labgpio_0.cpp
 *
 *  Created on: 14-Feb-2018
 *      Author: Kaustubh
 */

#include <labgpio.hpp>
#include "uart0_min.h"
#include "stdio.h"


LabGPIO_0::LabGPIO_0(uint8_t portNumber,uint8_t pinNumber)
{
  portNum = portNumber;
  pinNum = pinNumber;
}

void LabGPIO_0::setAsInput(void)
{
  if(portNum == 1)
  {
    LPC_GPIO1->FIODIR &= ~(1<<pinNum);
  }
  else if(portNum == 0)
  {
    LPC_GPIO0->FIODIR &= ~(1<<pinNum);
  }
}

void LabGPIO_0::setAsOutput(void)
{
  if(portNum == 1)
  {
    LPC_GPIO1->FIODIR |= (1<<pinNum);
  }
  else if(portNum == 0)
  {
    LPC_GPIO0->FIODIR |= (1<<pinNum);
  }
}

void LabGPIO_0::setDirection(bool op)
{
  if(op == 1)
```

```
    {
        setAsOutput();
    }
    else
    {
        setAsInput();
    }
}

void LabGPIO_0::setHigh(void)
{
    if(portNum == 1)
    {
        LPC_GPIO1->FIOCLR = (1<<pinNum);
    }
    else if(portNum == 0)
    {
        LPC_GPIO0->FIOCLR = (1<<pinNum);
    }
}

void LabGPIO_0::setLow(void)
{
    if(portNum == 1)
    {
        LPC_GPIO1->FIOSET= (1<<pinNum);
    }
    else if(portNum == 0)
    {
        LPC_GPIO0->FIOSET = (1<<pinNum);
    }

}

bool LabGPIO_0::getLevel(void)
{
    uint8_t z;
    if(portNum == 1)
    {
        if(LPC_GPIO1->FIOPIN & (1<<pinNum))
        {
                z=1;
        }
        else
        {
                 z=0;
        }
    }

    else if(portNum == 0)
    {
        if(LPC_GPIO0->FIOPIN & (1<<pinNum))
        {
                 z=1;
        }
        else
        {
            z=0;
        }
    }
    return z;
}


void LabGPIO_0::set(bool high)
{
    if(high)
    {
        setHigh();
    }
    else
    {
        setLow();
    }
}

LabGPIO_0::~LabGPIO_0(void)
{

}




Main.cpp
#include "FreeRTOS.h"
#include "task.h"
#include "uart0_min.h"
#include "LPC17xx.h"
#include "labgpio.hpp"
#include "stdio.h"
#include "utilities.h"

uint8_t global_int = 0;
uint8_t global_ext = 0;

void vControlLEDint( void * pvParameters )
{
    /* Get Parameter */
    uint32_t param = (uint32_t)(pvParameters);
    /* Define Constants Here */

    /* Define Local Variables and Objects */
    LabGPIO_0 obj(1,param);
    obj.setDirection(1);
    uint8_t led = 0;
   // w =obj.getLevel();
    /* Initialization Code */

    while(1)
    {
        /* Insert Loop Code */
        if(global_int == 1)
        {
            led=obj.getLevel();
            obj.set(!led);
            global_int=0;
            delay_ms(100);
        }
        obj.set(led);
    }
}

void vReadSwitchint( void * pvParameters )
{
    /* Get Parameter */
    uint32_t param = (uint32_t)(pvParameters);
    /* Define Constants Here */

    /* Define Local Variables and Objects */

    LabGPIO_0 obj1(1,param);
    obj1.setDirection(0);
    uint8_t local_switch = 0;

    /* Initialization Code */

    while(1)
    {
        /* Insert Loop Code */

        while(obj1.getLevel() == 1)
        {
            local_switch=1;
        }
        if(local_switch == 1)
        {
            global_int^=1;
        }
        local_switch=0;
        delay_ms(100);
    }
    /* Only necessary if above loop has a condition */
   // xTaskDelete(NULL);
}

void vControlLEDext( void * pvParameters )
{
    /* Get Parameter */
    uint32_t param = (uint32_t)(pvParameters);
    /* Define Constants Here */

    /* Define Local Variables and Objects */
    LabGPIO_0 objext(0,param);
        objext.setDirection(1);
        uint8_t ledext = 0;
        /* Initialization Code */

        while(1)
            {
                /* Insert Loop Code */
            if(global_ext == 1)
            {
                ledext=objext.getLevel();
                objext.set(!ledext);
                global_ext=0;
                delay_ms(100);
            }
            objext.set(ledext);
        }
}

void vReadSwitchext( void * pvParameters )
{
    /* Get Parameter */
    uint32_t param = (uint32_t)(pvParameters);
    /* Define Constants Here */

    /* Define Local Variables and Objects */

    LabGPIO_0 objext1(0,param);
        objext1.setDirection(0);
        uint8_t local_switch = 0;

        /* Initialization Code */

        while(1)
        {
            /* Insert Loop Code */

            while(objext1.getLevel() == 1)
            {
                local_switch=1;
            }
            if(local_switch == 1)
            {
                global_ext^=1;
            }
            local_switch=0;
            delay_ms(100);
        }
}

int main(int argc,const char* argv[])
{
    const uint32_t Size = 1024;

    LPC_PINCON->PINSEL0 &= ~((3<<1)|(3<<3));

    xTaskCreate(vReadSwitchint, (const char*)"SwitchReadTask",Size,(void*)9,1,NULL);
    xTaskCreate(vControlLEDint, (const char*)"LEDControlTask",Size,(void*)0,1,NULL);
    xTaskCreate(vReadSwitchext, (const char*)"SwitchReadTaskext",Size,(void*)0,1,NULL);
    xTaskCreate(vControlLEDext, (const char*)"LEDControlTaskext",Size,(void*)1,1,NULL);

    vTaskStartScheduler();

    return 0;
}
```