```
/*Assignment 6 - UART */
LabUART obj;
char out;
void vSendTask(void *p)
{
    while(1)
    {
        obj.switchIp();
    }
}

void vReceiveTask(void* p)
{
    while(1)
    {
        obj.receive();
    }
}

int main()
{
    while(1)
    {
        int x = obj.init(LabUART::UART2);
        LD.setLeftDigit(48);
        LD.setRightDigit(48);
        if(x==1)
        {
            xTaskCreate(vSendTask, (const
char*)"SendTask",1024,(void*)0,1,NULL);
            xTaskCreate(vReceiveTask, (const
char*)"ReceiveTask",1024,(void*)0,1,NULL);
            vTaskStartScheduler();
        }
    }
    return 0;
}
```

//LabUART.hpp

```
#ifndef LABUART_H
```

```
#define LABUART_H

#include "LPC17xx.h"
#include "stdio.h"


class LabUART
{
private:

    uint8_t uartPort;
public:

    enum UartModes
    {
        /* Fill this out based on the datasheet. */
        UART2 = 0,
        UART3 = 1,
    };

    LabUART();
    ~LabUART();

    // TODO: Fill in methods for init(), transmit(), receive() etc.
    //decides whether UART2 is used or UART3.
    //initializes PINSEL block, interrupts, clk and powers the UART2
and UART3
    bool init(UartModes num);

    //accepts inputs from switches
    void switchIp();

    //interrupt service routine for UART2
    static void my_isr2();

    //interrupt service routine for UART3
    static void my_isr3();

    //transmits the data to be transmitted(i/p from switch)
    void transmit(char data);

    //receives queued data and deque it to accept the received
character
    //compute result of the operation sent (2 operands and one
operator)
    char receive();

    // Optional: For the adventurous types, you may inherit from
"CharDev" class to get a lot of funcionality for free
```

```
};

#endif
```

//LabUART.cpp

```
#include "labUART.hpp"
#include "FreeRTOS.h"
#include "lpc_isr.h"
#include "queue.h"
#include "io.hpp"
#include "utilities.h"

QueueHandle_t handle = 0;
uint16_t br = (96000000 / (16 * 57600));
char a[3] = {};
int i=0;
char l,j,k=0;

LabUART::LabUART()
{}

bool LabUART::init(UartModes num)
{
    uartPort = num;

    if(uartPort == 0)
    {
        LPC_SC->PCONP &= ~(1<<24);  //power UART2
        LPC_SC->PCONP |= (1<<24);

        LPC_SC->PCLKSEL1 &= ~(3<<16);   //UART2 Clock select
        LPC_SC->PCLKSEL1 |= (3<<16);

        LPC_UART2->LCR |= (1<<7);    //DLAB=1

        LPC_UART2->DLL |= br & 0xFF;   //baud = 9600
        LPC_UART2->DLM |= br>>8;

        LPC_UART2->LCR &= ~(1<<7);  //DLAB=0

        LPC_UART2->IER |= (7<<0);   //RBR and THRE enable

        LPC_UART2->FCR |= ((1<<0)|(1<<2));   //FIFO reset
        LPC_UART2->FCR |= ((1<<0)|(1<<6));   //FIFO enable

        LPC_PINCON->PINSEL4 &= ~((3<<16)|(3<<18));
        LPC_PINCON->PINSEL4 |= ((2<<16)|(2<<18));   //UART2 Pin
selects
```

```
        isr_register(UART2_IRQn, my_isr2);  //register UART2
interrupt

        NVIC_EnableIRQ(UART2_IRQn);

        handle = xQueueCreate(4,sizeof(char));
    }

    else if(uartPort == 1)
    {
        LPC_SC->PCONP &= ~(1<<25);  //power UART3
        LPC_SC->PCONP |= (1<<25);

        LPC_SC->PCLKSEL1 &= ~(3<<18);   //UART3 Clock select
        LPC_SC->PCLKSEL1 |= (3<<18);

        LPC_UART3->LCR |= (1<<7);    //DLAB=1

        LPC_UART3->DLL |= br & 0xFF;   //baud = 57600
        LPC_UART3->DLM |= br>>8;

        LPC_UART3->LCR &= ~(1<<7);  //DLAB=0

        LPC_UART3->IER |= (7<<0);   //RBR and THRE enable

        LPC_UART3->FCR |= ((1<<0)|(1<<2));   //FIFO reset
        LPC_UART3->FCR |= ((1<<0)|(1<<6));   //FIFO enable

        LPC_PINCON->PINSEL9 &= ~((3<<24)|(3<<26));
        LPC_PINCON->PINSEL9 |= ((3<<24)|(3<<26));   //UART3 Pin
selects

        isr_register(UART3_IRQn, my_isr3);  //register UART3
interrupt

        NVIC_EnableIRQ(UART3_IRQn);

        handle = xQueueCreate(4,sizeof(char));
    }
    else
    {
        printf("Invalid port selection\n");
        return 0;
    }
    return 1;
}

void LabUART::switchIp()
```

```
{
    //char swState=0;

    if(SW.getSwitch(1)) //1st operand input
    {
        while(SW.getSwitch(1));
        l++;
        if(l>9)
            l=0;
        LD.setRightDigit(l+48);
        printf("1st switch i/p:%d\n",l);

        //swState = 1;
    }
    if(SW.getSwitch(2)) //2nd operand input
    {
        while(SW.getSwitch(2));
        j++;
        if(j>9)
            j=0;
        LD.setLeftDigit(j+48);
        printf("2nd switch i/p:%d\n",j);
    }

    if(SW.getSwitch(3)) //operator input
    {
        while(SW.getSwitch(3));
        k++;
        if(k>2)
            k=0;
        LD.setNumber(k);
        printf("3rd switch i/p:%d\n",k);
    }

    if(SW.getSwitch(4)) //transmit the 2 operands and 1 operator
    {
        transmit(l);
        delay_ms(1000);
        transmit(j);
        delay_ms(1000);
        if(k==0)
            transmit(10);
        if(k==1)
            transmit(11);
        if(k==2)
            transmit(13);
        delay_ms(1000);
    }
}
```

```
void LabUART::my_isr2(void)
{
        printf("in isr\n");
        char out = LPC_UART2->RBR;
        xQueueSend(handle,&out,1000);
        LPC_UART2->IIR |= (1<<0);
}
void LabUART::my_isr3(void)
{
        printf("in isr\n");
        char out = LPC_UART3->RBR;
        xQueueSend(handle,&out,1000);
        LPC_UART3->IIR |= (1<<0);
}

void LabUART::transmit(char data)
{
    if(uartPort == 0)
    {
        printf("in transmit\n");
        LPC_UART2->THR = data;
        while(!(LPC_UART2->LSR & (1<<6)));
    }
    else if(uartPort == 1)
    {
        printf("in transmit\n");
        LPC_UART3->THR = data;
        while(!(LPC_UART3->LSR & (1<<6)));
    }
}

char LabUART::receive()
{
    char out,c,d,x = 0;
    char b=19;
    if(xQueueReceive(handle,&out,500))
    {
        if(i==2)
        {
            a[i] = out;
            if(a[i]==10)
                d = '*';
            if(a[i]==11)
                d = '+';
            if(a[i]==13)
                d = '-';
            printf("received operator is:%c\n",d);
            i=0;
```

```
    if(a[2] == 10)
        c = a[0]*a[1];
    else if(a[2] == 11)
        c = a[0]+a[1];
    else if(a[2] == 13)
    {
        if(a[1]>a[0])
        {
            c = a[1] - a[0];
            x=1;
        }
        else
            c = a[0]-a[1];
    }
    else
        printf("invalid operation\n");

    printf("received data is : %d\n",c);
    LD.setNumber(c);
    if(x==1)
    {
        transmit(c%10);
        transmit(b);
    }
    else if(x==0)
    {
        transmit((c%10));
        transmit((c/10));
    }
    receive();
    receive();
    if(x==1)
    {
        LD.setRightDigit((c%10)+48);
        LD.setLeftDigit(b+48);
    }
    else if(x==0)
    {
        LD.setRightDigit((c%10)+48);
        LD.setLeftDigit((c/10)+48);
    }
    i=0;
    x=0;
}
else
{
    a[i] = out;
    if(i==0)
    {
```

```
                printf("a[0]:%d\n",a[i]);
                LD.setRightDigit(a[i]+48);
            }
            if(i==1)
            {
                printf("a[1]:%d\n",a[i]);
                LD.setLeftDigit(a[i]+48);
            }
            i++;
        }
    }
    return c;
}
LabUART::~LabUART()
{}
```