(a) (5) Write *all* required setup code such that all of the pins are inputs. Do not assume any bits have known values at power-on.

```
GPIOPinWrite (CTLSTAT, 0X02);      //enable port by setting pin1 high
GPIOPinWrite (DIR, 0x00);      //0 is input
```

1
10
311
100

-2

(b) (15) Let there be an empty array of 10 1-byte values declared as `uint8_t stuff[10]`. Write a polling loop that waits for individual bytes to arrive on the interface and then puts them into sequential slots in the array.

```
volatile uint8_t i=0;
while (i<10)
{ int8_t flagVal= GPIOPinRead (CTLSTAT, GPIO_PIN_3); //checks if in
  while (0x04 != flagVal)
  { flagVal = GPIOPinRead ( CTLSTAT, GPIO_PIN_3) ///loop until something is
  }
  stuff[i] = GPIOPinRead (DATA, 0x11111111);//store incoming byte in array
  i++; //increment array storage location
}
return 0;
```

-10

(c) (15) Write a complete ISR to replace your polling loop. Do not worry about any additional initializations or any interrupt enables (global or specific). *Only* write the complete ISR.

```
void IntGPIO(void)
{   if(i<10)
    { stuff[i] = GPIOPinRead( DATA, 0x11111111; //store incoming byte in array
      i++; //incr array slot
    }
    GPIOPinWrite(INTCTL, 0x01); // clear interrupt flag
}
return;
```

-10

-22