HAND IN

Answers recorded on exam paper

Student number								

QUEEN'S UNIVERSITY FINAL EXAMINATION

FACULTY OF ENGINEERING & APPLIED SCIENCE

APSC 142 – Introduction to Computer Programming for Engineers

M. Moussa (Sec. 100); T. Okpotse (Sec. 101); B. Lynch (Sec. 102)

April 18, 2016, 2:00 pm

INSTRUCTIONS TO STUDENTS:

1. This examination is 3 HOURS in length. **NO AIDS ARE ALLOWED.**
2. Please read all questions carefully and answer in the space provided on the examination paper.
3. Do not separate pages from this exam paper; **write your student number in the space provided at the top of each page**.
4. Please use scrap paper to draft your solutions, then copy your completed solutions neatly to the space provided on the exam paper. You may hand in your scrap paper, but it will not be marked.
5. Comments in your C code are not expected and will not be marked.
6. For full marks, your C code must be reasonably efficient as well as correct. Global variables are not permitted on this exam.

GOOD LUCK!

PLEASE NOTE: "Proctors are unable to respond to queries about the interpretation of exam questions. Do your best to answer exam questions as written.

Question 1 (20 marks)		Question 4 (12 marks)	
Question 2 (12 marks)		Question 5 (20 marks)	
Question 3 (20 marks)			
Marker Initials		**TOTAL (84 marks)**	

Student number:								

1. Program Comprehension (12 marks)

1.A (2 marks)

In the box below, show the screen output that would be produced by the following program:

```
task main()
{
      int x = 3;
      int y = 7;

      y = (2*x + 5)/x;
      displayTextLine(0, "y = %d", y);

      y = x*y;
      displayTextLine(1, "y = %d", y);
}
```

1.A

Student number:								

1.B (3 marks)

In the box below, show the screen output that would be produced by the following program:

```
task main()
{
     float xMid = 5.0;
     float xLow, xHigh;
     float x = 5.7;
     float w = 1.5;

     xLow = xMid - w;
     xHigh = xMid + w;

     displayTextLine(0,"xLow = %.1f", xLow);
     displayTextLine(1,"xHigh = %.1f", xHigh);

     if ((x < xLow) || (x > xHigh))
     {
          displayTextLine(3,"x Range Error");
     }
     else
     {

          displayTextLine(3,"No Error");

     }
}
```

1.B

Student number:								

1.C (3 marks)

In the box below, show the screen output that would be produced by the following program:

```
#define N 3

task main()
{
     int i;
     float z = 0.0;

     for (i = 0; i < N; i++)
     {
          z = z + 0.25*i;
          displayTextLine(i, "z = %.2f", z);
     }
}
```

1.C

Student number:								

1.D (5 marks)

In the box below, show the screen output that would be produced by the following program:

```
task main()
{
     int i;
     int xSet[] = {3, 1, 5, 0, 2};

     xSet[2] = xSet[3] - xSet[1];
     xSet[4] = xSet[3] - xSet[2];

     for (i = 0; i < 5; i++)
     {
          xSet[i] = 2*xSet[i];
          displayTextLine(i, "%d", xSet[i]);
     }
}
```

1.D

Student number:								

1.E (4 marks)

In the box below, show the screen output that would be produced by the following program:

```
task main()
{
      float a = 4.0;
      int j = 0;

      while (a < 10)
      {
            displayTextLine(j, "a = %.1f", a);

            a += 2.5;
            j++;
      }
      displayTextLine(j, "a = %.1f", a);
}
```

1.E

Student number:								

1.F (3 marks)

In the box below, show the screen output that would be produced by the following program:

```
int scramble(int x, int &y);

task main()
{
     int a = 4;
     int b = 5;
     int c = 7;

     c = scramble(a, b);

     displayTextLine(0, "a = %d", a);
     displayTextLine(1, "b = %d", b);
     displayTextLine(2, "c = %d", c);
}

int scramble(int x, int &y)
{
     y = x + y;
     return x;
}
```

1.F

Student number:								

2. Computation and Output (12 marks)

Given the following polynomial:

$$y = x^3 - 6x^2 + 11x - 6$$

Compute the value of y at evenly spaced x points starting at $x = 0$, incrementing by 0.25, and ending at $x = 5$ (i.e., $x = 0$, $x = 0.25$, $x = 0.5$, $x = 0.75$, ... $x = 5$) and *set the LED colour to red if the value of y is less than 0 otherwise set it to green*. Have the program *wait 500 ms* at the end of each computation.

Enter your code on the following page.

Student number:								

```
task main()
{
      // declare variables




      // compute each value of y and set the LED colour









}
```

Student number:								

3. Robot Operation (20 marks)

You are given an EV3 robot configured with the following sensors:

Port	Sensor Name	SensorType
S1	Touch (left)	`sensorEV3_Touch`
S2	Touch (right)	`sensorEV3_Touch`
S3	Ultrasonic	`sensorEV3_Ultrasonic`
S4	--	`--`

The *left motor* is plugged into *motor port B* and the *right motor* is plugged into *motor port C.*

Write a program that will navigate a corridor using the sensors to react to the walls according to the rules listed below. The program should *use a continuous loop* that reads the sensor measurements and then reacts accordingly at each loop iteration. The program should also *wait 100 ms at the end of each iteration.*

1. If the *left touch sensor is bumped* then command the *left motor to have speed 20* and *right motor to have speed -20,* and *set the LED colour to be green.*

2. If the *right touch sensor is bumped* then command the *left motor to have speed -20* and *the right motor have speed 20,* and *set the LED colour to be red.*

3. If *both touch sensors are bumped,* set *both wheel speeds to 0, call the function* `plotStopSign`, *wait 2000 ms,* and *end the program.*

4. If *neither touch sensors are bumped, set both wheel speeds to 40.*

5. At *each iteration, display the measured distance on the LCD* screen using `displayTextLine` with 2 decimal places in the format "`d = X.XX`" (any line number is acceptable).

You will also need to *complete the function* `plotStopSign` to *plot the stop sign image on the LCD screen.* Note that the image is simply plotted starting at the origin on the screen (i.e., no offset position).

Enter your code on the following 2 pages. Your code may extend into the second box if you need more space than provided in the first box. Note that the function should be completed in the third box.

Student number:								

```
void plotStopSign();

task main()
{
      // declare variables




      // set sensor types
      SensorType[S1] = sensorEV3_Touch;
      SensorType[S2] = sensorEV3_Touch;
      SensorType[S3] = sensorEV3_Ultrasonic;


      // continuous loop
      while (          )
      {
            // read sensors




            // command motors and LED colour based on given rules









            // display measured distance and wait 100 ms



      }
}
```

Student number:								

```
// extend your code into this extra space if needed
```

Student number:								

```
// complete the plotStopSign function:

void plotStopSign()
{
      int stopSignImage[8][8] = {{0,0,1,1,1,1,0,0},
                                 {0,1,1,1,1,1,1,0},
                                 {1,1,1,1,1,1,1,1},
                                 {1,1,1,1,1,1,1,1},
                                 {1,1,1,1,1,1,1,1},
                                 {1,1,1,1,1,1,1,1},
                                 {0,1,1,1,1,1,1,0},
                                 {0,0,1,1,1,1,0,0}};

      // declare loop counters
      int i, j;

      // plot each pixel
      for (                      )
      {
           for (                      )
           {




           }
      }
}
```

Student number:								

4. 1D Arrays (12 marks)

You are given a series of eight temperature measurements in the following list:

[11.1, 13.7, 12.8, 15.2, 17.9, 14.7, 18.1, 19.0]

Write a program that counts the number of values in the array that are greater than 15 and stores that value in the variable `warmDays`. Your program should also fill a second array named `warmTemps` with 1's corresponding to temperatures greater than 15, and 0's corresponding to temperatures less than or equal to 15 (the resulting array that should be filled is shown below for clarity):

[0, 0, 0, 1, 1, 0, 1, 1]

The program should also display the value of `warmDays` at the end of the program using `displayTextLine` and with the format "`Temps above 15: X`" (any line number is acceptable).

Complete the code on the following page.

Student number:								

```
#define N 8

task main()
{
      // declare variables









      // check each value

















      // display value of warmDays

}
```

Student number:								

5. Functions (20 marks)

You are given a robot with an ultrasonic distance sensor connected to sensor port 1 (S1) and a colour sensor connected to port 2 (S2) in default ambient light mode. The measured distance is used to set the LED colour and set the maximum robot speed according to the following:

Measured Distance	LED Colour	Maximum Speed
$d \geq 30$	green	60
$30 > d \geq 10$	orange	25
$d < 10$	red	0

The robot speed is set to be equal to the measured ambient light but also limited to never exceed the maximum speed *(i.e., if the measured ambient light is greater than the maximum speed, the robot speed is simply set to be the maximum speed)*. Note that the left and right wheels are connected to motor ports B and C, respectively.

Write a program to run in a continuous loop, reading the sensor values at each iteration and implementing the given rules using two functions:

`updateLED` – Takes as one input the measured distance, sets the LED colour, and returns the maximum speed as an output.

`updateSpeed` – Takes as the first input the measured ambient light and as a second input the maximum speed, sets the robot speed on both wheels to be equal to the ambient light measured value or the maximum speed (whichever is lower).

Enter your code on the following 2 pages.

Student number:								

```
// function declarations/prototypes
int updateLED(int distance);
void updateSpeed(int light, int maxSpeed);


task main()
{
      // declare variables




      // set up sensor types
      SensorType[S1] = sensorEV3_Ultrasonic;
      SensorType[S2] = sensorEV3_Color;


      // continuous loop


            // read sensor values
            dist = SensorValue[S1];          // distance
            ambLight = SensorValue[S2];      // ambient light


            // call functions to set LED colour and set speed









}
```

Student number:								

```
// functions
```

APSC 142 – Final Examination Information Sheets
C Functions and Program Operators and Structures

Robot C commands and built-in functions:

SensorType[] SensorMode[] sensorEV3_Color sensorEV3_Touch sensorEV3_Ultrasonic sensorEV3_Gyro sensorSound_DBA	abs(float val) cos(float radians) sin(float radians) sgn(float x) sqrt(float x) exp(float x) log(float x) log10(float x)	sleep(int msecs) wait1Msec(int msecs) random(long range)

```
setMotorSpeed(int motor, int speed)
getMotorRunning(int motor)
getMotorEncoder(int motor)
moveMotorTarget(int motor, int ticks, int speed)
waitUntilMotorStop(int motor)
getUSDistance(int port)
getColorName(int port)
getColorReflected(int port)
getTouchValue(int port)
playTone(int freq, byte 10msec_tics)
setPixel(int x, int y)
clearPixel(int x, int y)
displayTextLine(int line, " ", ...)
displayStringAt(int line, string " ", ...)
eraseDisplay()
```

Built-in colour codes and LED settings:

colorNone	colorYellow	ledOff	ledOrangeFlash
colorBlack	colorRed	ledRed	ledOrangePulse
colorBlue	colorWhite	ledRedFlash	ledGreen
colorGreen	colorBrown	ledRedPulse	ledGreenFlash
		ledOrange	ledGreenPulse

Operators:

- parentheses: ( ) - innermost first
- unary: + - ++ -- !
- binary: * / % + -
- relational binary: < <= > >= == !=
- logical binary: && ||
- assignment: = += -= *= /=

Repetition structures:

```
while (condition)
{
   ... statements;
}
```

```
do
{
  ... statements;
} while (condition);
```

```
for (exp_1; exp_2; exp_3)
{
  ... statements;
}
```

Selection structures:

```
if (condition)
{ ...  statments
when
  condition is true;
}
else
{ ... statements
when
  condition is
false;
}
```

```
if (condition_1)
{ block 1}
else if
(condition_2)
{ block 2}
.
.
else if
(condition_n)
{block n}
else
{block_of_code}
```

```
switch( variable )
case val_1:
{ statements;
  break; }
case val_2:
{ statements;
  break; }
.
.
case val_n:
{ statements;
  break; }
```

Arrays:

```
1D array: vals[N] = {vals[0], vals[1], ... vals[N-1]};
2D array: vals[N][M] = {{vals[0][0], vals[0][1], ... vals[0][M-1]},
                        {vals[1][0], vals[1][1], ... vals[1][M-1]},
                         ...
                        {vals[N][0], vals[N][1], ... vals[N][M-1]}};
```

Functions:

- function types - byte, int, float, void
- prototype: functionType functionName (parameters);
- function:

```
functionType functionName (parameters)
{
    block_of_code
      ...
      return variable; //not included for void function
}
```

- pass by value: e.g., int func(int num)
- pass by reference: e.g., void func(int &num)

Functions and arrays:

- typedef - 1D array: typedef type typeName[N];
- typedef - 2D array: typedef type typeName[ROWS][COLS];
- function prototype with array:
  - void functionName (typeName, other parameters);