

**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**

B. Lynch (Sec. 100); T. Karamat (Sec. 101); M. Karaim (Sec. 102)

April 24, 2017, 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 and/or the back of the pages 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.**

This material is copyrighted and is for the sole use of students registered in [the course] and writing this exam. This material shall not be distributed or disseminated. Failure to abide by these conditions is a breach of copyright and may also constitute a breach of academic integrity under the University Senate's Academic Integrity Policy Statement.

Question 1 (15 marks)		Question 4 (15 marks)	
Question 2 (15 marks)		Question 5 (15 marks)	
Question 3 (25 marks)		Question 6 (10 marks)	
<b>Marker Initials</b>		<b>TOTAL (95 marks)</b>	

Student number:								
--------------------	--	--	--	--	--	--	--	--

## 1. Program Comprehension (15 marks)

### 1.A (5 marks)

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

```
task main()
{
    int x = 3;
    int y = 5;
    float z;

    x = x*y + 1.7;
    displayTextLine(0, "x = %d", x);

    z = x - y;
    displayTextLine(1, "z = %.3f", z);

    y = x % 3;
    z += 2*y;
    displayTextLine(2, "z = %.5f", z);

    z = (x + y/4.0);
    displayTextLine(3, "z = %.3f", z);

    y = !(z > x);
    displayTextLine(4, "y = %d", y);
}
```

1.A	
Line 0	
Line 1	
Line 2	
Line 3	
Line 4	
Line 5	
Line 6	

Student number:								
--------------------	--	--	--	--	--	--	--	--

### 1.B (5 marks)

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

```
task main()
{
    int i;
    int count = 10;

    for (int i = 0; i <= 2; i++)
    {
        count += i;
        displayTextLine(i, "value %d is %d", i+1, count);
    }

    if (count > 13)
    {
        displayTextLine(5, "high");
    }
    else
    {
        displayTextLine(5, "low");
    }
    displayTextLine(6, "final output = %.2f", 0.5*count);
}
```

1.B	
Line 0	
Line 1	
Line 2	
Line 3	
Line 4	
Line 5	
Line 6	

Student number:								
--------------------	--	--	--	--	--	--	--	--

### 1.C (5 marks)

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

```

float newFunction(float a, float b);

task main()
{
    int i;
    float set[] = {4.3, 2.1, 5.0, 1.3};
    float x, newSet[3];

    set[1] = 3.3;
    set[3] = set[1] - set[3];

    for (int i = 0; i <= 2; i++)
    {
        newSet[i] = newFunction(set[i], set[i+1], x);
        displayTextLine(i, "%.1f", newSet[i]);
    }
    displayTextLine(5, "%.1f", x);
}

float newFunction(float a, float b, float &d)
{
    float c;
    c = a + b;
    d = a - b;
    return c;
}

```

1.B	
Line 0	
Line 1	
Line 2	
Line 3	
Line 4	
Line 5	
Line 6	

Student number:								
--------------------	--	--	--	--	--	--	--	--

## 2. Computation and Output (15 marks)

Write a program to do the following:

- Compute the values of the following polynomial function  $y = f(x)$  between  $x = 0$  and  $x = 2$  stepping by increments of 0.2 (i.e.,  $x = 0.0, 0.2, 0.4, \dots, 1.6, 1.8, 2.0$ ).

$$y = 4x^3 + 3x^2 + 2x - 1$$

- Find the *maximum* value of  $y$  in that range and output it on the screen with 3 decimal places.
- Store the values of the polynomial in an array called `yData`.
- Compute the sum of the values in the array `yData` using a separate `for` or `while` loop and output it on the screen with 3 decimal places.

Enter your code on the following page.

Student number:								
--------------------	--	--	--	--	--	--	--	--

```
task main()
{
```

```
}
```

Student number:							
--------------------	--	--	--	--	--	--	--

### 3. Robot Operation (25 marks)

Write a program for the EV3 robot that will react to obstacles and coloured tiles using the touch sensors, colour sensor, and ultrasonic distance sensor according to the following requirements:

1. Set the LED colour to flashing red and keep waiting until the robot is placed on a green tile. Once placed on a green tile, the robot should start driving forward at a speed setting of 40 and the LED colour should be set to green.
2. While driving, the robot should turn left 90 degrees if only the right touch sensor is activated or turn right 90 degrees if only the left touch sensor is activated (*assume that you have access to the functions void leftTurn() and void rightTurn() that make the appropriate turns*).
3. While driving, the robot speed should be set to 20 if the ultrasonic distance sensor measures a distance less than 50 cm, set to 60 if the ultrasonic distance sensor measures a distance greater than 100 cm, otherwise set to 40.
4. While driving, if the robot measures a yellow tile using the colour sensor, the LED colour should be set to orange, if the robot measures a red tile, the LED colour should be set to red, otherwise the LED colour should be set to green.
5. While driving, if both touch sensors are activated or if the colour sensor measures a blue tile, the robot should stop, the LED colour should be set to flashing red, the program should wait 3 seconds and then end.

Include the following functions in your program:

`void bumpTurn()` – has no inputs and no outputs, but checks the left and right touch sensors and causes the robot to turn if necessary using `leftTurn()` or `rightTurn()` (according to requirement #2 above).

`int distanceSpeed(int distance)` – takes the measured ultrasonic distance as input and returns the required robot speed (according to requirement #3 above).

`void colourToLED(int colourCode)` – takes the measured colour sensor colour code as input and sets the robot LED colour (according to requirement #4 above).

The left and right motors are plugged into motor ports B and C, respectively. The following table lists the connected sensors. Enter your code on the following pages.

Port	Sensor Name	Sensor Type
S1	Touch	<code>sensorEV3_Touch</code>
S2	Touch	<code>sensorEV3_Touch</code>
S3	Ultrasonic	<code>sensorEV3_Ultrasonic</code>
S4	Colour	<code>sensorEV3_Color</code>

Student number:								
--------------------	--	--	--	--	--	--	--	--

```
void leftTurn();  
void rightTurn();  
void bumpTurn();  
int distanceSpeed(int distance);  
void colourToLED(int colourCode);  
  
task main()  
{  
  
    // set sensor types (fill in the rest):  
    SensorType[S1] = sensorEV3_Touch;  
  
    SensorMode[S4] = modeEV3Color_Color;  
    // keep checking the colour sensor before driving:  
  
    // keep driving...  
    while (true)  
    {  
  
    }  
}
```

Student number:								
--------------------	--	--	--	--	--	--	--	--

```
// functions:  
  
void leftTurn()  
{  
    setMotorSpeed(motorB,-30);  
    setMotorSpeed(motorC,30);  
    wait1Msec(750);  
    setMotorSpeed(motorB,0);  
    setMotorSpeed(motorC,0);  
}  
  
void rightTurn()  
{  
    setMotorSpeed(motorB,30);  
    setMotorSpeed(motorC,-30);  
    wait1Msec(750);  
    setMotorSpeed(motorB,0);  
    setMotorSpeed(motorC,0);  
}
```

Student number:								
--------------------	--	--	--	--	--	--	--	--

// extra space if necessary

Student number:								
--------------------	--	--	--	--	--	--	--	--

#### 4. Function and 1D Arrays (15 marks)

Write a program that analyzes a given array `inputData` according to the following requirements:

1. Computes the differences between subsequent values in the array `inputData` and stores the results in a new array `diffData`
2. Finds the average difference (i.e., the average of the values in `diffData`)
3. Counts the number of positive values in `diffData`

Your program must use a single function to accomplish the above requirements (outputting the array `diffData`, the average difference, and the number of positive values). Your program must also display the average difference and number of positive values in `diffData` according to the following format:

Average difference: X.XXXX  
Number of positive values: X

Note that the array `diffData` will have one fewer element than the array `inputData`. Use the following values for the array `inputData` (the corresponding values that would result in `diffData` are also shown for clarity):

`inputData: [3.2, 6.9, 7.2, 6.4, 4.7, 5.1, 5.5, 3.8, 2.3, 3.0]`

`diffData: [3.7, 0.3, -0.8, -1.7, 0.4, 0.4, -1.7, -1.5, 0.7]`

Complete the code on the following pages.

Student number:								
--------------------	--	--	--	--	--	--	--	--

```
task main()
{
    // declare and initialize variables:

    // call function:

    // output results:

}

// function definition:
```

Student number:								
--------------------	--	--	--	--	--	--	--	--

// extra space if necessary

Student number:									
--------------------	--	--	--	--	--	--	--	--	--

## 5. 2D Arrays (15 marks)

Write a program that analyzes a given 2D array `dataTable` according to the following requirements:

1. Computes the sum of the elements in each row.
2. Computes the product of the elements in each column.
3. Counts the total number of elements less than five.

Assume you are given the following values for the 2D array `dataTable`:

3	1	7
6	4	5
8	9	2

The results should be output on the screen in the following format:

```
Row 0 sum: 11
Row 1 sum: 15
Row 2 sum: 19
Column 0 product: 144
Column 1 product: 36
Column 2 product: 70
# of elements < 5: 4
```

*Note that you are not required to use functions but may do so if you prefer.*

Enter your code on the following 2 pages.

Student number:								
--------------------	--	--	--	--	--	--	--	--

```
#define N 3
```

```
task main()
{
```

```
}
```

Student number:								
--------------------	--	--	--	--	--	--	--	--

```
// extra space if necessary
```

Student number:								
--------------------	--	--	--	--	--	--	--	--

## 6. Algorithms (10 marks)

Complete the following program to determine the unique values in an array `rawData` and store them in a new array `uniqueData`. The array `uniqueData` is initialized with the same size as `rawData` and containing all -1 elements, and should be filled with the unique values in `rawData` with no -1 values in between.

For example, the following values for the array `rawData` will result in the array `uniqueData` as shown below.

```
rawData = {5, 3, 2, 5, 7, 3, 9, 3, 2, 7};  
uniqueData = {5, 3, 2, 7, 9, -1, -1, -1, -1, -1};
```

Enter your code on the following 2 pages.

Hints:

- Use the loop counter `k` to track the number of unique values and fill the array `uniqueData` without skipping elements.
- Use the variable `repeat` to track if a value is repeated, if so then do not add it to `uniqueData`.

Student number:									
--------------------	--	--	--	--	--	--	--	--	--

```
#define N 10

task main()
{
    // declare and initialize variables
    int i, j, k = 0;
    int rawData[N] = {5, 3, 2, 5, 7, 3, 9, 3, 2, 7};
    int uniqueData[N];
    int repeat;

    // set uniqueData array to start as -1 for each element
    for (i = 0; i < N; i++)
        uniqueData[i] = -1;

    // output results
    for (i = 0; i < k; i++)
        displayTextLine(2*i, "%d", unique[i]);
}
```

Student number:								
--------------------	--	--	--	--	--	--	--	--

//extra space if necessary

## APSC 142 – Final Examination Information Sheets

### C Functions and Program Operators and Structures

#### Robot C commands and built-in functions:

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

#### 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) { ... statements 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);
- definition:  

```
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);