Student number:								

HAND IN
answers recorded on question paper

Queen's University
Faculty of Engineering and Applied Science

APSC 142 - Introduction to Computer Programming for Engineers

FINAL EXAMINATION – 20 April 2013, 9:00am

Instructors: Dr. Ahmad Afsahi, Mr. Basel Nabulsi, and Dr. Evelyn Morin

Exam duration is 3 hours.

This is a closed book exam. NO AIDS ARE ALLOWED. Information sheets are provided at the end of the exam paper.

Read all questions carefully and answer in the space provided.

Please do not separate pages from this exam booklet. Write your student number at the top of each page.

Please use scrap paper to draft your solutions, then copy your completed solutions, neatly, to the exam paper in the space provided for each question. (You may hand in your rough work, but it will not be marked.)

Comments in your C code are not expected and will not be considered in marking.

For full marks, C code must be reasonably efficient as well as correct. Global variables are NOT allowed on this exam.

This examination will count for 55% or 65% of your final course grade,

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

For instructor use only:

Question 1:	/12	Question 4:	/16
Question 2:	/14	Question 5:	/18
Question 3:	/16	Question 6:	/18
		Total Marks:	/94

Student Number:	

1. Program Comprehension (12 marks)

a) The code below was written to compute the average values of two x variables and two y variables. In the box beneath the code, show the screen output.

```
task main ()
{
  int x1 = 5, x2 = 8, y1 = 14, y2 = 7;
  float x_average, y_average;
    x_average = x1 + x2 / 2;
    y_average = y1 + y2 / 2.0;
  nxtDisplayTextLine (0, "x-average is %0.2f", x_average);
  nxtDisplayTextLine (1, "y-average is %0.2f", y_average);
}
```

Does the program compute the correct average values for x and y? _____

If not, write the correct expressions for x_average and y_average below.

b) For the code on the following page, complete the function `findMax` to find the maximum value in the array nums. The parameters to be passed to the function include the array, array size, and the variable `max` which should be passed by reference.

Student Number:	

```
typedef int Int1D[5];
void findMax (Int1D, int, int &);
task main()
{
    int nums[5] = {2, 8, 5, 9, 4};
    int max;
      max = nums[0];
    findMax (nums, 5, max);
    nxtDisplayTextLine(0, "Max is: %d", max);
}

void findMax (
{
```

c) For the following code, show the screen output in the box below.

```
task main()
{
    int i, j;
    int vals[3][3] = { {20, -6, -12},
                       {15, 12, -30},
                       {-14, 9, 11}};
    int sums[3];

    for (i=0; i<=2; i++)
    {
        sums[i] = vals[i][0];
        for (j=1; j<=2; j++)
            sums[i] = sums[i] + vals[i][j];
        if (sums[i] < 0)
            sums[i] = 0;

        nxtDisplayTextLine (i, "Term [%d]: %d", i, sums[i]);
    }
}
```

Student Number:	

2. Computation and Output (14 marks)

Write a program that calculates values of the polynomials

$$y_1 = 8x^3 - 20x^2 + 75x - 25$$

$$y_2 = 5x^3 - 25x^2 - 50x + 75$$

at $x = 0$, $x = 0.25$, $x = 0.5$, …, $x = 5.0$ (that is, for values of x separated by 0.25 between a start value of 0 and an end value of 5.0). Your program should determine the largest absolute difference (abs(y2 – y1)), *max_diff*, and the smallest absolute difference, *min_diff*, between the corresponding values of the two polynomials for the x-values in the range [0, 5], and then display them.

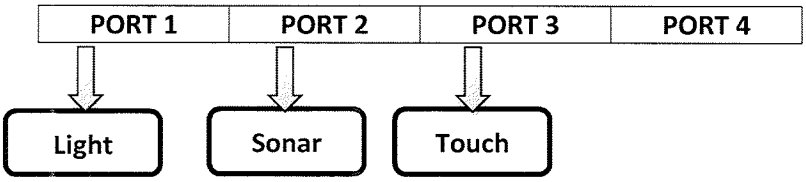
Complete the **main()** in this box.

Student Number:	

3. Robot Programming (16 marks)

A robot is placed at point A in the maze shown below. The maze is bounded by walls and the width of all the pathways is the same and is equal to 100 cm. At the end of the maze (point B), there is a black tape on the floor to indicate the end of the maze.

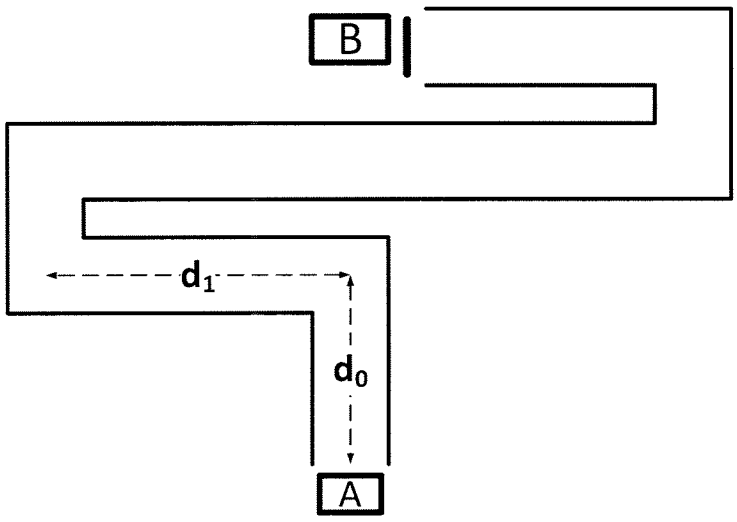
The robot has 4 ports as shown below, with sensors attached as shown:



Your task is summarized as follows:

- Navigate the robot from point **A** to point **B** using any combination of sensors you might feel suitable for the task.
- As the robot travels through the maze, it will store the length of each segment $\mathbf{d_i}$ travelled in a 1-D array. For example, if the robot travels the maze shown below, a valid reading could be: {143, 285, 90, ... } . (The diameter of the wheels of the robot is 3 cm)
- Upon recording the distance $\mathbf{d_i}$ travelled in segment **i,** the robot will play a tone with frequency **1000 Hz** for a duration of $\mathbf{d_i}$ milliseconds.
- The robot will make a full stop upon reaching the black tape on the ground near the point **B**. The light sensor reading for the black tape is in the range 25 to 40. The light sensor reading for the white ground is in the range 60 to 80.

Enter your code on the next page. The function `Rotate` is included on the following page.



Student Number:	

```
#define DIAMETER 3
Rotate(int speed, int degrees);
task main()
{
    //GIVEN DECLARATIONS
    int lightval, sonardist, index, toneduration;
    float d[10], circumference;
    // attach the sensors to the corresponding ports
```

```
void Rotate(int speed, int degrees)
{
     int rot_msecs; //how many milliseconds I need to rotate
     float fval;   //to avoid integer overflow

     fval =  (210.0 * abs(degrees))/speed;
     rot_msecs =  (int) fval; //back to 16-bit integer size
     if(degrees>0)
     {    motor[1] = +speed;
          motor[2] = -speed;
     }
     else
     {    motor[1] = -speed;
          motor[2] = +speed;
     }
     //wait for needed rotation time
     wait1Msec(rot_msecs);

     //set motors back to 0 speed
     motor[1] = 0;
     motor[2] = 0;
     wait1Msec(200);
}
```

Student Number:	

4. 1D Array (16 marks)

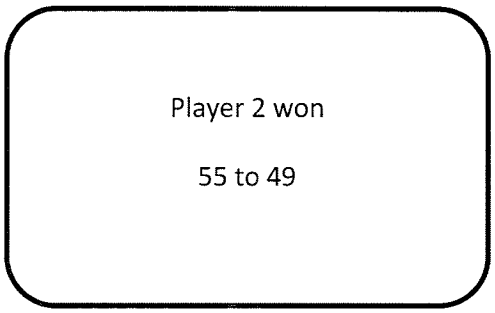
You are asked to design a simple card game using RobotC. The game is played by two players. Each player starts with a set of 52 cards numbered [1 to 13] four consecutive times

Deck 1={1,2,3,...,12,13,1,2,3,...,12,13,1,2,3,...,12,13,1,2,3,...,12,13}

Deck 2={1,2,3,...,12,13,1,2,3,...,12,13,1,2,3,...,12,13,1,2,3,...,12,13}

Description of the game:

- Each player randomly selects a card from their own respective deck.
- Then the cards are compared with each other and the difference between the values of the cards is recorded.
- The player with the higher card gains 2 points.
- If the cards are equal, each player gains 1 point only.
- Upon playing a card from a deck, the place of the card should be filled in with a zero, and that card cannot be played again.
- When both players finish playing 52 moves, the number of cards left in their decks is zero.
- You should display which player won the game on the 4th line and the score on the fifth line of the LCD display. For example, if we run the game and player 2 wins, the display should look like:



To generate a random selection, you will use the RobotC function **`random()`**. This function, if called with an argument `i` will generate a random number between 0 and `i-1`.

Enter your code on the next page.

Student Number:	

```
task main()
{

    int Deck1[52]={1,2,3,4,5,6,7,8,9,10,11,12,13,
                   1,2,3,4,5,6,7,8,9,10,11,12,13,
                   1,2,3,4,5,6,7,8,9,10,11,12,13,
                   1,2,3,4,5,6,7,8,9,10,11,12,13};

    int Deck2[52]={1,2,3,4,5,6,7,8,9,10,11,12,13,
                   1,2,3,4,5,6,7,8,9,10,11,12,13,
                   1,2,3,4,5,6,7,8,9,10,11,12,13,
                   1,2,3,4,5,6,7,8,9,10,11,12,13};
```

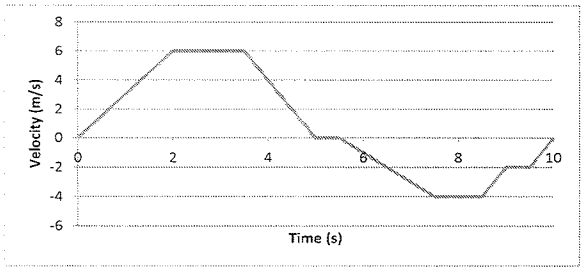
Student Number:	

5. Computation and Numerical Methods (18 marks)

An autonomous vehicle travels along a rail in a straight line. Starting at time, t = 0 s, the velocity of the vehicle is measured at 0.5 s intervals for 10 s. The measured values are given in the table below, where positive values represent velocity in the forward direction, and negative values represent velocity in the backward direction.

Time (s)	0.0	0.5	1.0	1.5	2.0	2.5	3.0	3.5	4.0	4.5	5.0
Velocity (m/s)	0.0	1.5	3.0	4.5	6.0	6.0	6.0	6.0	4.0	2.0	0.0
Time (s)	5.5	6.0	6.5	7.0	7.5	8.0	8.5	9.0	9.5	10.0	
Velocity (m/s)	0.0	−1.0	−2.0	−3.0	−4.0	−4.0	−4.0	−2.0	−2.0	0.0	

The time-velocity curve for the vehicle is :



The acceleration of the vehicle is the change in velocity with the change in time : $\frac{\Delta v}{\Delta t}$.

The displacement, or distance travelled over any time interval is the area under the time velocity curve.

Complete the C program on the following page to:

- calculate the acceleration of the vehicle in each 0.5s segment and store the acceleration values in an array in `task main ()`
- determine the final displacement of the vehicle from the starting point, where the displacement is the distance travelled forwards minus the distance travelled backwards; to do this :
    - call a function – `areaUnderCurve` – that performs Riemann sum integration over the time-velocity curve
        - the width of each rectangle will be the sample period, 0.5s
        - the height of each rectangle will be the mid-point value of the velocity in the sample period
    - return the result of the integration to `task main ()`

Student Number:	

```
typedef float Array1D[21];

float areaUnderCurve (Array1D, float, int);

task main()
{
        float velocities[21] = {0.0, 1.5, 3.0, 4.5, 6.0, 6.0,
                                6.0, 6.0, 4.0, 2.0, 0.0, 0.0,
                               -1.0, -2.0, -3.0, -4.0, -4.0,
                               -4.0, -2.0, -2.0, 0.0};









float areaUnderCurve (
```

Student Number:	

6. 2D Array and Function (18 marks)

The two-dimensional array, `courseMarks`, contains information about 700 students in a first year course. The array contains the student numbers, lab marks, quiz marks, project marks, and the final exam marks in the first column, second column, third column, fourth column, and the fifth column, respectively. You are to write a program that uses the *Selection Sort* algorithm to sort the rows of the array in ascending order according to the student number in the first column. A small example of the array before and after sorting is shown below.

Before Sorting

Student number	Lab	Quiz	Project	Final exam
23556	8	15	18	51
20754	10	13	20	49
29261	7	12	19	53
16765	9	14	19	47

After Sorting

Student number	Lab	Quiz	Project	Final exam
16765	9	14	19	47
20754	10	13	20	49
23556	8	15	18	51
29261	7	12	19	53

Your main program in **`task main()`** calls the function **`selectionSort()`** to sort the 2-D array using the **`swapRows()`** function, and then displays the sorted array. The `selectionSort()` function accepts as parameters a 2-D array, an integer representing the number of rows in the 2-D array, and an integer representing the index of the column in the 2-D array by which the array is to be sorted. The `swapRows()` function accepts as parameters a 2-D array, two integers indicating the row indices of the array that will be swapped, and an integer representing the number of columns in the array.

Hint: In the `selectionSort()` function, perform the Selection Sort algorithm as in 1-D array case by looping through the elements of the first column of the array courseMarks. In the swapRows() function, use a **for** loop to swap all the elements of the two rows.

Enter your code in the boxes on the following pages.

Student Number:	

Complete `main()` in the box below.

```
#define ROWS 700
#define COLS 5

typedef int Int2D[700][5];

void selectionSort (Int2D, int, int);
void swapRows (int, int, int, Int2D);

task main()
{
    int courseMarks[ROWS][COLS] = {{23556, 8, 15, 18, 51},
                                   {20754, 10, 13, 20, 49},
                                   {29261, 7, 12, 19, 53},
                                   {16765, 9, 14, 19, 47}
                                        ...          };
                          // 2D-array fully initialized
```

Complete the definition (body) of **selectionSort()** in this box.

```
void selectionSort (Int2D nums, int row, int col)
{
```

Student Number:	

Complete the definition (body) of **swapRows()** in this box.

```
void swapRows (int row1, int row2, int col, Int2D arrayNums)
{
```

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

Robot C commands and built-in functions:

motor[] SensorType[] SensorValue[] sensorTouch sensorSONAR sensorLightActive sensorSoundDBA	abs(float val) cos(float radians) sin(float radians) sgn(float x) sqrt(float x) exp(float x)	wait1Msec(int msecs) wait10Msec(int msecs) random(long range) log(float x) log10(float x)
PlayTone(int freq, byte 10msec_tics nxtSetPixel(int x, int y) nxtClearPixel(int x, int y) nxtDisplayTextLine(int line, " ", ...) nxtDisplayString(int line, string quoted_str) nxtDisplayStringAt(int x, int y, string quoted_str) eraseDisplay()		

Operators:

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

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; }
```

Repetition structures:

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

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 fund(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);