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

Dec 17, 2016, 9:00 am

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.
7. Except on questions for which robot C is indicated, write your code in standard C.

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 (12 marks)		Question 4 (12 marks)	
Question 2 (12 marks)		Question 5 (12 marks)	
Question 3 (12 marks)		Question 6 (12 marks)	
		Question 7 (6 marks)	
Marker Initials		TOTAL (78 marks)	

Student number:								

1. Program Comprehension (12 marks)

1.A (4 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 + 5.0)/x;
     displayTextLine(1, "y = %d", y);

     y = x * y;
     displayTextLine(3, "y = %d", y);
}
```

1.B (4 marks)

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

```
task main()
{
     int i = 0, j = 3;

     do {
        displayTextLine(i, "%d", j );
        i++;
        j = j + i;
     } while (j < 15) ;
}
```

Student number:								

1.C (4 marks)

The function below is supposed to arrange the two passed-in integer variables, swapping them if necessary so that upon return, val1 is not less than than val2. If the values were swapped, then the function returns the value 1; otherwise, it returns the value 0.

In the box below, write the lines of C code that will complete the function.

```
int swap(int &val1, int &val2)
{
      [                                        ]
      return swapped;
}
```

Student number:								

2. Computation and Output (12 marks)

Write a program that calculates values of the polynomial $5x^3 + 2x^2-15x+3$ at x=0, x=0.2, x=0.4, ... , x=5.0 (that is, for values of x spaced apart by 0.2 between a start value of 0 and an end value of 3.0). Your program should then display the minimum value of the polynomial in that same range [0, 3], as well as the value of x at which the minimum occurs.

```
task main()
{




}
```

Student number:								

3. Robot Operation (12 marks)

A robot is placed at a point 200 cm away from a wall. The robot sonar sensor is connected to port 1, and the touch (bumper) sensors are attached to ports 2 and 3 and the sound sensor is connected to port 4. The left motor is connected to port A and the right motor is connected to port B.

Complete the code on the following two pages to do the following:

- Drive the robot forward towards the wall at full speed.
- Using the sonar sensor, continuously display the distance from the wall.
- Continuously display the current sound level.
- When the robot reaches a distance of 40 cm from the wall, reduce the speed to 20% of maximum speed
- If either of the touch sensors bumps into the wall, stop the robot and display the message "Ooops" and stop displaying the distance and sound levels.

Note: the code to check the bumpers must be placed in a separate function on the second page of code. Use the constants defined in your code.

Student number:								

```
#define FULL_SPEED 100

// function prototypes
int readTouchSensor();

task main() {

     // GIVEN DECLARATIONS
     int bump = 0; // stores return value of readTouchSensor
     int distance;

     //Initialize the sensor types



     // Continuously read the sonar and touch sensors (the
     // function readTouchSensor) and control the speed
    while(                          )
    {









     }
     // Stop the robot and display Ooops.



}
```

Student number:								

```
// This function reads the touch sensors. If either sensor bumps
// into an object, this function will return 1. If neither touch
// sensor is pressed, this function will return 0

int readTouchSensor()
{









}
```

Student number:								

4. 1-D 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 first calculates the average temperature and displays the average temperature on line 2 of the robot display. The program must also then count the number of values that are greater than the average and the number of values that are less than the average. These counts are to be displayed on lines 4 and 6 of the robot display.

Complete the code on the following page:

Student number:								

```
task main {
  // declare variables



  // initialize appropriate variables



  // compute average temperature



  // count the number of samples greater than and the number less than the average temperature



  // display average and temperature counts

}
```

Student number:								

5. 2-D Arrays (12 marks) `

Write a program and two function called flipArrayHorizontal() and flipArrayVertical() to swap the values in a 4x4 integer array and store it in another array. For example, if the input array contains:

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

The output of the flipArrayHorizontal() function is:

4	3	2	1
8	7	6	6
12	11	10	9
16	15	14	13

and the output of the flipArrayVertical() is:

13	14	15	16
9	10	11	12
5	6	7	8
1	2	3	4

Write the code in the boxes on the following pages.

Student number:								

Complete **main()** in this box:

```
typedef int Int2D[4][4];
// declare function prototypes



task main()
{
   // declare vararіables
   int firstArray[4][4] ={
     {  1,  2,  3,  4},
     {  5,  6,  7,  8},
     {  9, 10, 11, 12},
     { 13, 14, 15, 16};
   int flippedHorizontalArray[4][4];
   int flippedVerticalArray[4][4];



   // Call the functions here.




}
```

Student number:								

Complete the definition (body) of **flipArrayHorizontal()** in this box

```
void flipArrayHorizontal (Int2D array1, Int2D array2)
{



}
```

Student number:								

Complete the definition (body) of **flipArrayVertical()** in this box

```
void flipArrayVertical (Int2D array1, Int2D array2)
{




}
```

Student number:								

6. Simulating a Physical Problem (12 marks)

A rubber ball is dropped off the observation deck of the CN tower on a windless day. Accounting for air resistance and gravity, the ball undergoes a **downward** vertical acceleration given by $a = (9.8 - 0.021v^2)$ where the ball's **downward** velocity v is in units of meters per second.

On the following page, write a main program that models the flight of the rubber ball until it reaches the ground. Your program should use the small-time-step simulation approach where you advance in same time increments of $\Delta t = 0.01$ seconds, and do repeated computations to find the new downward velocity and vertical position at the end of each time **step until the ball's height drops below 0 (or equals 0).**

For each time step, your program should therefore:

1) Compute the corresponding velocity increase $\Delta v = (9.8 - 0.021v^2) * \Delta t$
2) Compute the final velocity at the end of the time step, $v = vprev + \Delta v$
3) Compute the average velocity over the time step
4) Compute the height change $\Delta h = vavg * \Delta t$, and the new height at the end of the time step

The final step of the program is to display the time that the ground was struck.

DO NOT use any functions in your solution. Use the defined constants in your solution.

Student number:								

Complete your solution to question 6 in this box:

```
#define HINIT 346 // CN observation deck height (meters)
#define DELTAT 0.01 // time step delta (secs)

task main()
{
    // YOUR DECLARATIONS



    // INITIALIZATION



    // FLIGHT LOOP



   // OUTPUT



}
```

Student number:								

6. Design Thinking (6 marks)

6.A (3 marks) Personas are an important part of the design thinking process. Give an example of a persona.

6.B (3 marks) A Minimum Viable Product is an important test of the economic feasibility of a product. Give an example of a minimum viable product.

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

Robot C commands and built-in functions:

SensorType[]	abs(float val)	abs(float val)
SensorMode[]	cos(float radians)	cos(float radians)
sensorEV3_Color	sin(float radians)	sin(float radians)
sensorEV3_Touch	sgn(float x)	sgn(float x)
sensorEV3_Ultrasonic	sqrt(float x)	sqrt(float x)
sensorEV3_Gyro	exp(float x)	exp(float x)
sensorSound_DBA	log(float x)	log(float x)
	log10(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()
```

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
{ statementswhen
 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: int func(int num)
- pass by reference: 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);