= COMP1917 Final Exam Parts ABC : 2014s1 =

== Total number of marks: 90+1 ==

== Total duration: 90 minutes ==

There is one mark for following the examination instructions.

Your answers can be submitted by pressing save on this application.

You may submit your solutions as many times as you like. The last submission ONLY will be marked.

Write your name on the top of each sheet of rough working paper you use, this will not be marked, and write your answers in the computer files as directed in the exam instructions. You will be provided with an electronic copy of the textbook.

You must hand in ALL writing paper at the end of the exam.

Once the exam has commenced you may not leave until the end of the exam (first session only).

Calculators may not be used. Compilers or other software tools may not be used. You may only use the specified editor and pdf viewer. The eating of toast in the exam room is prohibited. You must not attempt to communicate with any other person, or access other computers or any internet resources.

If you do not follow these instructions you will get zero marks for the exam and a possible charge of academic misconduct.

Unless advised otherwise you may not use any language features or library functions not covered in class. Code which does not comply with the course style guide, which contains a security vulnerability, or which is overly complex or unclear will be penalised.

Phones must be turned off and then wrapped in foil or paperso that the screen is not visible and then sealed in a plactic bag and placed under the seat. Students are permitted to bring at most one (1) balloon into the exam room provided this does not cause disruption to other students.

Questions (and sub-questions) are not worth equal marks. Answer all questions.

== PART A: Short Answer Questions ==

Your working is not marked in Part A. For multiple choice questions select the response which best answers the question. Keep your written answers clear and very brief. Messy or long answers will not be marked.

There are no Sympathy Marks in Part A.

=== Question 0 ===

(2 Marks)

If a dependent child is a person under 18 years of age who does not earn $10,000 or more a year, which expression would define a dependent child?

```
 .[A] (age < 18) &amp;&amp; (salary < 10000)
 .[B] (age < 18) || (salary < 10000)
 .[C] (age <= 18) &amp;&amp; (salary <= 10000)
 .[D] (age <= 18) || (salary <= 10000)
```

=== Question 1 ===

(2 Marks)

What are the values of girls, boys, and children after the following code has been executed?

```
{{{
    int girls = 0;
    int boys = 0;
    int children = 0;
    children = girls + boys;
    girls = 15;
    boys = 12;
}}}
```

 .[A] 0, 0, 0
 .[B] 0, 0, 27
 .[C] 15, 12, 0
 .[D] 15, 12, 27

=== Question 2 ===

(3 Marks)

Write code to shift the values in these variables around so that a is given b's original value, b is given c's original value, and c is given a's original value.

The following diagram illustrates the direction of the shifts:

{{diagram1.png}}

Don't write a sub-function to do this - just do it directly a few lines of code.

____
____
____
____
____

=== Question 3 ===

(3 Marks)

What will be the value of the variable z after the following code is executed?

```
{{{
int x = 1;
int y = 2;
int z = 3;
if (x < y) {
    if (y > 4) {
        z = 5;
    } else {
        z = 6;
    }
}
}}}
```

The value of z will be __________

=== Question 4 ===

(3 Marks)

Consider the following block of code, where variables a, b, c, and answer each store integer values:

```
{{{
    if (a > b) {
        if (b > c) {
            answer = c;
        } else {
            answer = b;
        }
    } else if (a > c) {
        answer = c;
    } else {
        answer = a;
    }
}}}
```

Which one of the following sets of values for a, b, and c will cause answer to be assigned the value in variable b?

 .[A] a = 1, b = 2, c = 3

 .[B] a = 1, b = 3, c = 2

 .[C] a = 2, b = 1, c = 3

 .[D] a = 3, b = 2, c = 1

=== Question 5 ===

(3 Marks)

What will be the value of result after the following code statements are executed?

```
{{{
    int nums1[LENGTH] = { 1, -5, 2, 0, 4, 2, -3 };
    int nums2[LENGTH] = { 1, -5, 2, 4, 4, 2, 7 };
    int result = 0;
    int j = 0;
    while (j < LENGTH) {
        if (nums1[j] != nums2[j]) {
            result = result + 1;
        }
        j = j + 1;
    }
}}}
```

The value of result will be __________

=== Question 6 ===

(3 Marks)

What is the purpose or outcome of the following piece of code?

```
{{{
   int number[LENGTH];
   ...
   int result = 0;
   int j = 0;
   while ( j < LENGTH) {
       if (number[j] < 0) {
           result = result + 1;
       }
       j++;
    }
}}}
```

.[A] To find the smallest number in the array

.[B] To count the negative numbers in the array

.[C] To sum the negative numbers in the array

.[D] To add 1 to each of the negative numbers in the array

.[E] To find the index of the first negative number in the array

=== Question 7 ===

(3 Marks)

What is the outcome or likely purpose of the following piece of code? Express your answer as a short phrase, like the phrases provided as possible answers in Question 6 above.

```
{{{
int result = 0;
int count = 1;
while (count <= num) {
    result = result + count;
    count++;
}
}}}
```

____

____

=== Question 8 ===

(4 Marks)

We can represent an array of integers as a sequence of elements arranged from left to right, with the first element at the left and the last element at the right.

Using this representation, a programmer wishes to move all elements of an array one place to the right, with the rightmost element being â€˜wrapped aroundâ€™ to the leftmost position, as shown in this diagram:

{{diagram2.png}}

Here is the code that performs that shift for an array

```
{{{
   int values[LENGTH];
}}}
```

referred to by the name values:

```
{{{
   int oldRight = values[LENGTH - 1];
   int j = LENGTH - 1;
   while (j > 0) {
       values[j] = values[j - 1];
       j--;
   }
   values[0] = oldRight;
}}}
```

For example, if values initially contains the integers [1, 2, 3, 4, 5], once the code has executed it would contain [5, 1, 2, 3, 4].

Write code that will undo the effect of the above code. That is, write code that will move all the elements of the array one place to the left, with the leftmost element being wrapped around to the rightmost position.

____
____
____
____
____
____
____
____
____

=== Question 9 ===

(4 Marks)

Write a function that will be passed an array of integers, and the number of elements in the array, and will calculate and return (as a double) the mean (average) of all the integers in the array.

____
____
____
____
____
____
____
____
____

== PART B: Mama Bear ==

Your working is not marked in Part B. Keep your answers clear and very brief. Messy or long answers will not be marked.

If you cannot answer a question write "1 Sympathy Mark Please" as the answer and you will be awarded one mark for the question. If there are multiple sub-questions in a question just write it as your answer for one of the sub-questions and then you will be awarded one mark for the entire question and no sub-questions will be marked.

=== Question 10 ===

(6 Marks)

The following function is supposed to print out a square made of asterisks as illustrated below:

```
{{{
drawSquare (5)
*****
*   *
*   *
*   *
*****

drawSquare (3)
***
* *
***

drawSquare (1)
*
}}}
```

You may assume that the side length supplied is always an odd number. However the code below has two bugs which prevent it from printing the desired squares correctly.

```
{{{
void drawSquare (unsigned int size) {
    assert ((size%2) != 0);  // size must be odd
    int row = 0;
    int col = 0;
    while (row < size) {
        while (col < size) {
            char draw;
            if ((row==0)||(row==(size-1))) {
               draw = '*';
            } else if ((col==0)||(col==(size-1))) {
               draw = '*';
            } else {
               draw = ' ';
            }
            printf ("%c",draw);
            col++;
        }
        row++;
    }
    printf ("\n");
}
}}}
```

Briefly state the first bug

____
____
____

Briefly state the second bug

____
____
____

=== Question 11 ===

(6 Marks)

The following code is compiled and runs without error on one of the CSE 32 bit lab machines. What does each printf display?

```
{{{
    typedef unsigned char byte;

    unsigned int nines = 999;
    byte * ptr = (byte *) &nines;


    printf ("%x\n",nines);
    printf ("%x\n",nines * 0x10);
    printf ("%d\n",ptr[0]);
    printf ("%d\n",ptr[1]);
    printf ("%d\n",ptr[2]);
    printf ("%d\n",ptr[3]);
}}}
```

`printf ("%x\n", nines);`

____

`printf ("%x\n", nines * 0x10);`

____

`printf ("%d\n", ptr[0]);`

____

`printf ("%d\n", ptr[1]);`

____

`printf ("%d\n", ptr[2]);`

____

`printf ("%d\n", ptr[3]);`

____

=== Question 12 : Buggles ===

(6 Marks)

The following function is given the number of days since the start of 1980 and is supposed to work out what the year is. However it has a serious (and famous) bug.

```
{{{
int currentYear (int days) {
    int year = 1980;
    while (days > 365) {
        if (isLeapYear(year)) {
            if (days > 366) {
                days -= 366;
                year++;
            }
        } else {
            days -= 365;
            year++;
        }
    }
    return (year);
}

}}}
```

Briefly state what the bug is:

____
____
____

=== Question 13 : Trace ===

(6 Marks)

Consider the following function which compiles and runs without error:

```
{{{
int turtle (int gamera, int achilles) {
    printf ("call (%d,%d)\n", gamera, achilles);
    while (gamera != 0) {
        if (achilles > 0) {
            achilles = turtle (gamera, achilles-1);
        } else {
            achilles = 1;
        }
        gamera--;
    }
    return (achilles+1);
}
}}}
```

What value is returned if you call turtle (2,1)?
____

=== Question 14 : 4004 ===

(6 Marks)

Consider the following 4004 program:

```
{{{{
10 15  3  2
 9 10  4  2
 8  2  5  0
 0  0  0  0
}}}}
```

The number in cell 1 is the input to the program (15 in the above example). What is the relationship between the the value stored in cell 1 and the value in r0 when the program finishes executing?

State your answer as a simple sentence eg "when it terminates this program has made R0 be twice the value in cell 1"

____

== PART C: Papa Bear ==

Your working is not marked in Part C. Keep your answers clear and very brief. Messy or long answers will not be marked.

If you cannot answer a question write "1 Sympathy Mark Please" as the answer and you will be awarded one mark for the question. If there are multiple sub-questions in a question just write it as your answer for one of the sub-questions and then you will be awarded one mark for the entire question and no sub-questions will be marked.

=== Question 15 ===

(10 Marks)

Consider the following code to encode a message using the Caesar cipher:

```
{{{
01 // encode a plain text message using the Caesar cipher
02 // ie replace a -> b,A->B,b->c,B->C...z->a,Z->A
03 // and leave punctuation etc unchanged
04 #include "encoder.h"
05
06 #define LAST_ENCODE "Z";
07 typedef char *string;
08
09 char * encode (string plainText) {
10    string encodeTable;
11    int i;
12
13    // create the encode/decode table
14    while (i<LAST_ENCODE) {
15       if ((i='z')||(i='Z'))
16          // wrap around and encode z->a, Z->A
17          encodeTable[i]=i-26;
18       encodeTable[i] = ++i;
19    }
20
21    /* now create the encoded message
22    char cipherText[MAX_MESSAGE_LENGTH];
23    i=0;
24    while (plainText[i] != 0) {
25        char plainTextLetter = plainText[i];
26        cipherText[i] = encodeTable[plainTextLetter];
27        i++;
28    }
29    return cipherText;
30 }
}}}
```

There are between 5 and 15 bugs in this code. Below indicate the bugs you find by writing them next to the corresponding line numbers. If a bug is a security vulnerability mark it with ##.

____

____

____

____

____

____

____

____

____

____

____

____

____

____

____

____

____

____

____

____

____

____

____

____

____

____
____
____
____
____

== Frozen!TM ==
Both questions 16 and 17 relate to the following game.

The game Frozen!TM involves 3 players (red,green,blue) taking turns dropping colored balls down a row of 8 vertical glass tubes.

Each tube can hold up to 6 balls.

The red player selects a tube and drops a red ball into it, then the green player selects a tube and drops a green ball into it, then the blue player selects a tube and drops a blue ball into it, then the red player selects a tube and drops a red ball into it, and so on, and on, until someone wins.

The precise rules for determining who has won can change from game to game but are always based on the arrangement and colors of the balls which have been played.

Below is a picture of a game after the following moves:

```
{{{
   Red: 4
   Green: 4
   Blue: 3
   Red: 5
   Green: 3
}}}
```

{{Frozen44353.png}}

=== Question 16 ===
== game.h ==
(10 Marks)

Your friend is writing an AI to play the game Frozen!TM which will implement a function:

```
{{{
int bestTube (Game g);
}}}
```

which returns the best tube (0..7) for the current player to drop their ball into.

Another friend is also writing a runGame function which will display the game, then ask player AIs for their move in turn, then make the move on their behalf, and then repeat until it computes that the game is over.

Your job is to design and implement an ADT called Game which they will both use.   Make it as simple as possible as you don't have much time to write it.

Write the Game.h file. Put in brief comments to explain
your interface.

____
____
____
____
____
____
____

____
____
____
____
____
____
____
____
____
____

=== Question 17 ===

(10 Marks)

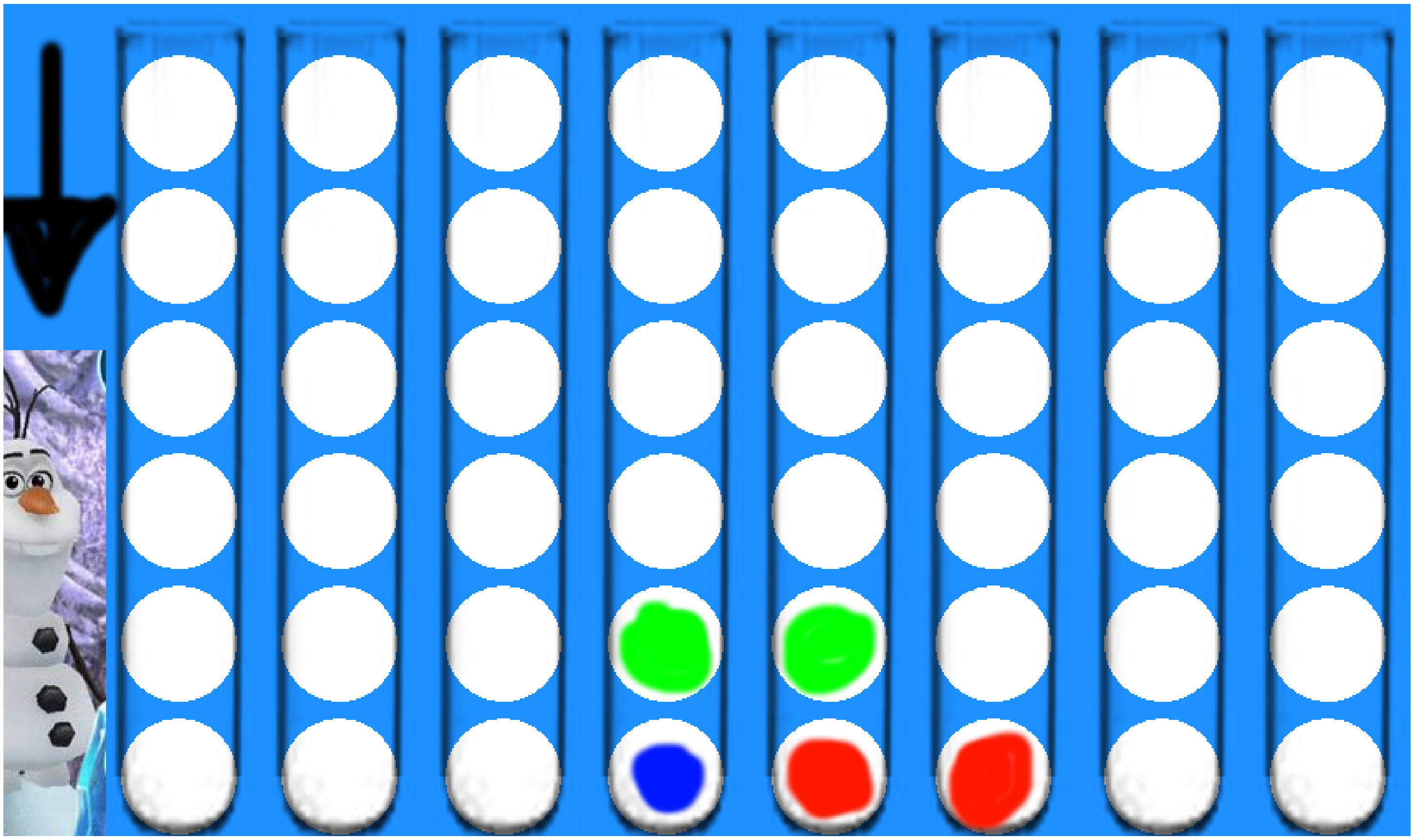
This question follows on from Question 16.

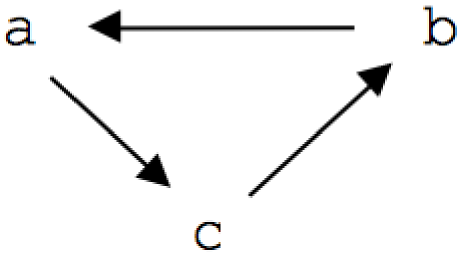
Write the Game.c file. Include brief comments where needed for clarity.

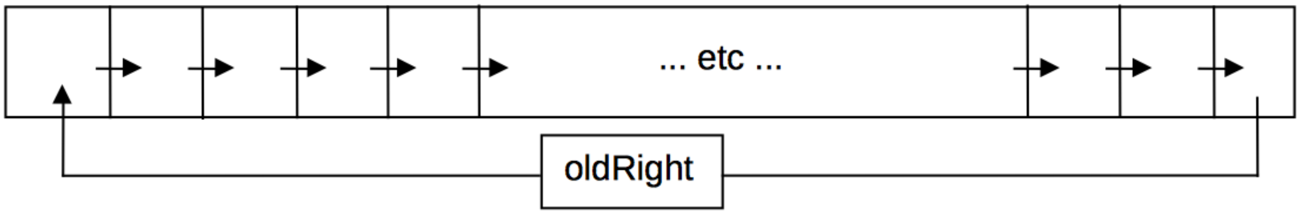
If you have more than 5 functions in your interface you need only implement 5 of them. In that case pick the 5 most important. Most important here means functions that it is essential the ADT provides rather than additional helper functions you might implement to help your friends and save them some coding.

____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____

IMAGE 1






... etc ...
oldRight