COMP1602: Computer Programming II
Course Work Exam #2
Duration: 1 ½ hours
Answer all questions.

1) A game is played on a board which is stored in a two dimensional (2D) array. The 2D array to represent the game board has a size of *numRows* and *numCols* where $0 \leq numRows < 100$, $0 \leq numCols < 100$, and $numRows = numCols$. A *Cell* struct stores the following data for **each** cell in the 2D array:
   - o Does the cell contain a light bulb?
   - o Is the cell hidden?
   - o The amount of immediate neighbours that contain a light bulb.

   (a) Declare a struct, *Cell,* to store information on each cell and declare the array, *board*, to store the game board. [2 marks]

   (b) Write a function, *initBoard*, which initializes the array such that each cell:
   - o Does not contain a light bulb
   - o Is hidden
   - o Has zero neighbours which contain light bulbs

   The prototype of *initBoard* is as follows:

```
void initBoard (Cell board[][100], int numRows);
```
   [4 marks]

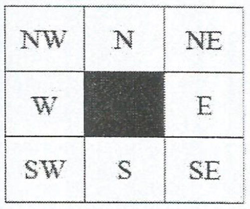
   (c) Write a function, *isValid*, which given a particular location (row, column), determines if the location is valid for the given board. A location is valid if it is within the range of the rows and columns of the initialized board. The prototype of *isValid* is as follows:

```
bool isValid (Cell board[][100], int numRows, int row, int col);
```
   [3 marks]

   (d) Write a function, *placeBulbs*, to randomly assign 10 percent of all the locations in the board with light bulbs. Note that every location on the board has two components, a row number and a column number. The prototype of *placeBulbs* is as follows:

```
void placeBulbs (Cell board[][100], int numRows);
```
   [5 marks]

   (e) A neighbour of a cell (shown in black on the diagram) is a cell that is North (N), South (S), East (E), West (W), North East (NE), South East (SE), South West (SW) or North West (NW) of the given cell, as shown in the diagram.



   One or more neighbours may not exist if the cell in on the edge of the board.

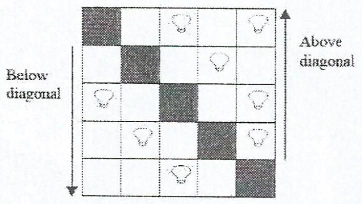
   Write a function, *getNumber*, which given the 2D board and the location of a particular cell, finds the amount of neighbours of the cell which contain a light bulb. You can use functions previously written.

   The prototype of *getNumber* is as follows:

```
int getNumber (Cell board[][100], int numRows, int row, int col);
```
   [6 marks]

   (f) The game board is said to be *crooked* if there are more light bulbs in the cells above the main diagonal than in the cells below the main diagonal. For example, the game board shown on the following page is crooked.

Write a function, *isCrooked*, which determines if the game board is crooked. The prototype of *isCrooked* is as follows:



```
bool isCrooked (Cell board[][100], int numRows);
```

[5 marks]

Total Marks: 25

2) A sorted integer array contains the following values:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
22	25	42	47	47	50	62	65	71	74	80	82	83	87	88	93	97

(a) A *binary search* is used to search for the value 88 in the array. How many searches will be made before it is found? Show your working. [4 marks]

(b) Suppose that a *linear search* is made.

i) How many searches will be made to find 88? / [1 mark]

ii) Does a linear search perform better on a sorted array or an unsorted array? Explain your answer. [2 marks]

Sorted

(c) Suppose a modified binary search function returns the *location* where a value should be inserted in a sorted integer array, if it is not present. The prototype of the modified *binary search* function is as follows:

```
int modBinarySearch (int A[], int numeElements, int key);
```

Write a segment of code which inserts a value *key* in the sorted integer array *A*. The number of elements in the array is *numElements*. Assume that there is enough space to accommodate the new value *key*. [4 marks]

Total Marks: 11

3) (a) Write a function, *startsWith*, which accepts two C-strings *s* and *t* as parameters and returns *true* if the contents of *s* start with all the contents of *t*, and *false* otherwise. For example: `startsWith("heaven", "he")` returns *true* but `startsWith ("heaven", "she")` returns *false*. The prototype of *startsWith* is as follows:

```
bool startsWith (char s[], char t[]);
```

You may use the *strlen* function from the C-string library. [4 marks]

(b) Write a function, *doubleLetter*, which accepts a C-string *s* and a character *t* as parameters and returns *true* if there are two consecutive occurrences of the character *t* in *s*, and *false* otherwise. For example: `doubleLetter ("hello", 't')` returns *true* and `doubleLetter ("cattle", 'c')` returns *false*. The prototype of *doubleLetter* is as follows:

```
bool doubleLetter (char s[], char t);
```

[5 marks]

Total Marks: 9

End of Test (Total Marks is 45)