Q1a: Read the Scratch code blocks shown below and answer the corresponding questions.  

When 'Green flag' clicked  
go to x: -100 y: -100  
wait 5 secs  
repeat until ([x position = 0] and [y position = 0]) {  
  wait 3 secs  
  change x by 10  
  change y by 10  
}

From the moment the green flag is clicked, how many seconds does the sprite take to reach (0,0)?  
Answer: 35 seconds  
Marking scheme: 1 Mark for correct answer, 0 otherwise. No partial marks.

When 'space' key pressed  
point in direction 0  
turn clockwise 15 degrees  
turn clockwise 15 degrees  
point in direction -90  
turn clockwise 15 degrees  
turn clockwise 15 degrees  
turn clockwise 15 degrees

In what direction is the Sprite pointing when this code has finished executing?  
Answer: -45 (Note the minus sign)  
Marking scheme: 1 Mark for correct answer, 0 otherwise. No partial marks.

When 'Down arrow' key pressed  
repeat until [length of names < 4] {  
say item 1 of names for 1 secs  
delete 1 of names  
}

What will be the result of executing this code when the list names contains entries as shown below?  

<table>
<thead>
<tr>
<th>Index</th>
<th>Item</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>2</td>
<td>B</td>
</tr>
<tr>
<td>3</td>
<td>C</td>
</tr>
<tr>
<td>4</td>
<td>D</td>
</tr>
<tr>
<td>5</td>
<td>E</td>
</tr>
</tbody>
</table>

Answer: A, B  
(In other words, first output is A, and second output is B)  
Marking scheme: 1 Mark for correct answer, 0 otherwise. No partial marks.

Refer to the code immediately above  
After the above result, what will be the output if you press the 'Down arrow' key again?  
Answer: There will not be any output.  
Marking scheme: 1 Mark for correct answer, 0 otherwise. Reason: A and B have been deleted, and length is now 3, so repeat until loop is not executed.
Q1b: Definition of Perrin numbers is: \( P(0) = 3, P(1) = 0, P(2) = 2, \) and \( P(n) = P(n - 2) + P(n - 3) \) for \( n > 2 \). The C++ program shown below takes two numbers \( a \) and \( b \) as input, and counts how many perrin numbers are odd and how many are even, in the range \([a, b]\) (including \( a \) and \( b \)). What will be the output when the input is \([9, 99]\), i.e., \( a \) is given as 9 and \( b \) is given as 99. [6 Marks]

```cpp
#include<iostream>
using namespace std;

int main(){
    int perrin[50];
    int low, high;
    int index=2, even=0, odd=0;
    cout << "Enter the values of a and b: ";
    cin >> low >> high;
    cout << endl;
    if (low >= 0 && low < high) {
        perrin[0] = 3; perrin[1] = 0; perrin[2] = 2;
        while (perrin[index] <= high) {
            if (perrin[index] >= low) {
                cout << perrin[index] << endl;
                if (perrin[index]%2 == 0) even++;
                else odd++;
            }
            index++;
            perrin[index] = perrin[index-2] + perrin[index-3];
        }
        cout<<"Even count is: ": even << endl;
        cout << "Odd count is: ": odd << endl;
    } else cout<<"Range given is not correct"; return 0;
}
```

Show the complete output of the program here:

Show the values of all the variables for the first three iterations of the while-loop here.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Before 1st iteration</th>
<th>After 1st iteration</th>
<th>After 2nd iteration</th>
<th>After 3rd iteration</th>
</tr>
</thead>
<tbody>
<tr>
<td>index</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>low</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>high</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>99</td>
</tr>
<tr>
<td>even</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>odd</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>perrin[0]</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>perrin[1]</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>perrin[2]</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>perrin[3]</td>
<td>-</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>perrin[4]</td>
<td>-</td>
<td>-</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>perrin[5]</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>5</td>
</tr>
<tr>
<td>perrin[6]</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Compile and run midsem-perrinNum.cpp, to verify.

Show the complete output of the program here:

Output:

<table>
<thead>
<tr>
<th>Even count is: 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Odd count is: 4</td>
</tr>
</tbody>
</table>

Marking scheme:
1st iteration [1 Mark]: Correct values shown for all variables upto perrin[3], as above. Zero marks if any value shown is wrong. 0.5 marks if values shown are correct, but all the variables are not shown.

2nd iteration [1 Mark]: Correct values shown for index and perrin[4]. It is ok if the other variables are not shown.

3rd iteration [1 Mark]: Correct values shown for index and perrin[0] to perrin[5]. It is ok if the other variables are not shown.

Output numbers [1 Mark]: Correct values output, in the order as shown. No partial marks.

Even count [1 Mark]: Correct value output, as shown. No partial marks.

Odd count [1 Mark]: Correct value output, as shown. No partial marks.
Q2a: Write a Scratch program to generate the honeycomb pattern shown below. All the hexagons are identical, and all the sides of each hexagon are of equal length. [10 Marks]

Run midsem-honeycomb.sb, to see working. Code shown below.

Marking scheme:
(Note to TAs: Some students may have different solutions from the one shown above. This is ok, if their logic is valid. Check their logic and give marks accordingly. Some students may have written only pseudo-code, instead of scratch program. These are also to be evaluated.)

If only pseudo-code is given – Divide the Marks by 2 (for each step below).

Initialization [1 Mark] – ‘clear’ and ‘pen down’ instructions are shown. It is ok if ‘go to (0, 0)’ is not shown.

Inner loop [4 Marks] – Correct number of iterations, move instruction, and correct degree of rotation in turn. It is ok if the number of steps moved or turn direction are different, as long as they are consistent.

Outer loop [4 Marks] – Correct number of iterations, move instruction, and correct degree of rotation in turn. Clarity [1 Mark] – 1 Mark if code is correct and logic of the code can be easily followed, 0.5 Marks if code is correct but difficult to follow, 0 Marks if code is incorrect.
Q2b: Consider two Scratch threads. We shall refer to one of them as **producer** and the other as **consumer**. Both threads start when Green flag is clicked. They share an array (list) called **buffer**. The producer thread’s job is to keep generating a new item of data and put it into the buffer. At the same time, consumer thread’s job is to keep removing data from the buffer, one item at a time.

Suppose that the maximum capacity of the buffer is N items. The producer should not try to add data into the buffer if the buffer is already full, and the consumer should not try to remove data from an empty buffer. In other words, if the buffer is full (has N items), the producer has to wait till some items have been removed, before it continues. Similarly, if the buffer is empty (has 0 items), the consumer has to wait till some items have been added, before it continues.

Write a Scratch program to implement this producer-consumer problem. Assume that: (i) N is 10, (ii) the data items generated by the producer are numbers, in an increasing sequence, 1, 2, 3, ..., and (iii) the consumer simply outputs each data item that it removes from the buffer. [10 Marks]

Run midsem-buffer.sb, to see working. Code shown below.

![Producer Thread](image1.png)

![Consumer Thread](image2.png)

Marking scheme:
(Note to TAs: Some students may have different solutions from the one shown above. This is ok, if their logic is valid. Check their logic and give marks accordingly. Some students may haee written only pseudo-code, instead of scratch program. These are also to be evaluated.)

If only pseudo-code is given – **Divide the Marks by 2** (for each step below).

- **Initialization** [1 Mark] – buffer and variables are initialized correctly.
- **Producer** [3 Marks] – Infinite loop and buffer full condition check are shown correctly.
  - Insert item [1 Mark] – Item generated and inserted correctly into buffer.
- **Consumer** [3 Marks] – Infinite loop and buffer empty condition check are shown correctly.
  - Insert item [1 Mark] – Item displayed and deleted correctly from buffer.
- **Clarity** [1 Mark] – 1 Mark if code is correct and logic of the code can be easily followed, 0.5 Marks if code is correct but difficult to follow, 0 Marks if code is incorrect.
Q3a: Write a C++ program to read a word from the input, and output its reverse. For example, if the input is 'midsem.', the output should be 'mesdim'. Assume that the word does not have blank spaces and the word ends when a fullstop (character .) is input. You may not use any libraries other than <iostream>. [10 Marks]

Compile and run midsem-reverseWord.cpp, to see working. Code shown below.

```cpp
#include<iostream>
using namespace std;

int main() {
    char arr[30]; // Assume that no word has more than 30 characters
    int i = 0, length;

    cout << "Give the word: ";
    while (1) {
        cin >> arr[i]; // cin will ignore blankspace, so check for .'
        if (arr[i] == '.') {length = i; break;}
        else i++;
    }

    cout << endl;
    for (i = length-1; i >= 0; i--) cout << arr[i]; //Assume that '.' is not to be output
    cout << endl;
    return 0;
}
```

Marking scheme:
(Note to TAs: Some students may have different solutions from the one shown above. This is ok, if their logic is valid. Check their logic and give marks accordingly. Some students may hae written only pseudo-code, instead of scratch program. These are also to be evaluated.)

If only pseudo-code is given – Divide the Marks by 2 (for each step below).

Initialization [1 Mark] – Reasonable assumption is made regarding size of char array.
Reading input [4 Marks] – Loop setup is correct and array entries are inserted correctly.
Printing output [4 Mark] – Loop setup and output are done correctly. Deduct 1 Mark if the '.' at the end of the word is also output.
Clarity [1 Mark] – 1 Mark at TA’s discretion on readability of the code.
Q3b: Write a C++ program that given an integer \( n \), generates a sequence \( R \) as follows: 
\( R_1 = \text{product of the digits of } n, \ R_2 = \text{product of the digits of } R_1, \) and so on, till some \( R_i \) becomes a single digit.

For example, if \( n = 999 \), then \( R_1 = 729, \ R_2 = 126, \ R_3 = 12, \) and \( R_4 = 2 \). So your program should output 729, 126, 12, 2, and terminate. If \( n = 6725 \), your program should output 420, 0, and terminate. [10 Marks]

Compile and run midsem-numSequence.cpp, to see working. Code shown below.

```cpp
#include <iostream>
using namespace std;

int main() {
    int num, d[10], newN;
    int size, i;

cout << "Give a number (upto 10 digits): "; cin >> num;
do {
    cout << "The number is: " << num << endl;
    i = 0;
    while (num > 0) {
        d[i] = num % 10; i++;
        num /= 10;
    }
    size = i;
    newN = 1;
    for (i = 0; i < size; i++) newN = newN * d[i];
    cout << "The product of its digits is: " << newN << endl;
    num = newN;
} while (newN/10 > 0);
return 0;
}
```

Marking Scheme:
(Note to TAs: Some students may have different solutions from the one shown above. This is ok, if their logic is valid. Check their logic and give marks accordingly. Some students may have written only pseudo-code, instead of scratch program. These are also to be evaluated.)

If only pseudo-code is given – Divide the Marks by 2 (for each step below).

Initialization [1 Mark] – Reasonable assumption is made regarding size of integer array.
Read input [1 Mark] – Number read correctly.

While loop [2 Mark] – Correct extraction of digits of current number into the array.
For loop [2 Mark] – Correct calculation of new number as product of digits.
Do While loop [2 Mark] – Correct outer loop to repeat the steps for the new number, till single digit.

Output [1 Mark] – Each number is output correctly.
Clarity [1 Mark] – 1 Mark at TA's discretion on readability of the code.