Quiz - 1 Section B

iv) 5 marks

Solution & Grading Scheme

Quiz 1

> Section B

Question 1 - 5 marks

Problem Description

Write a program to find the sum of first `n` even numbers. Here `n` can be taken to be an input from the user.

Solution

Both Codes are considered to be valid.

```
#include<stdio.h>
int main()
{
    int i, ans = 0,n;
    scanf("%d", &n);
    for(i=0; i < n; i ++)
    {
        ans = ans + 2*i;
    }
    printf("%d\n", ans);
    return 0;
}
```

```
#include<stdio.h>
int main()
{
    int i, ans = 0,n;
    scanf("%d", &n);
    int ans = n*(n-1); // n*(n+1) is also given as correct ans.
    printf("%d\n",ans);
    return 0;
}
```

Grading Scheme

1 mark: For writing

```
#include <stdio.h>
```

1 mark: For writing

```
int main(){
    return 0
}
```

3 marks: For correctness of the code

Question 2 - 5 marks

Problem Description

Explain the control flow between the RAM, Hard disk and the CPU after a program is written.

Solution & Grading Scheme

There are a few different interpretations to the question. The question expects you to explain what happens when a program is run.

The OS-kernel loads the program from the Hard disk into the RAM. **(1 mark)** It is then transmitted to the CPU through the memory bus. **(1 mark)** The CPU performs the following cycle: **(2 marks)**

1. Fetch
2. Decode
3. Execute After this the output is written back to the RAM through the memory bus. **(1 mark)**

Other interpretations include:

1. What happens when the program is written in the editor and saved?
2. What happens when the program is compiled?

Marks have been given to these interpretations if points similar to what we were looking for in the intended solution are given but according to the different interpretations.

Question 3 - 5 marks

Problem Description

Write `-15` in 1’s complement form (assume `8-bit` representation)

Solution & Grading Scheme

The number 15 in binary is given by: `00001111` **(2 marks)**
In 1’s complement it is given by: `11110000` . **(3 marks)**

Question 4 - 5 marks

Problem Description

State True/False with justification: Given two numbers `x` and `y`, the expression `y ^ ((x ^ y)& - (x < y))` calculates the minimum of the two numbers.

Solution

True, This expression does calculate the minimum of the two numbers **(1 Mark)**

Justification:

```
If x < y (2 Marks)
(x < y) = 1

-(x<y) = -1

-(x<y) = (1111...1) (Due to one’s compliment representation of negative numbers

(x ^ y) & (1111...1)_2 = (x ^ y) (As 1 is the identity for AND operation)

y^(( x ^ y) & - (x < y) = y ^ (x  ^ y) = x
y ^ (x ^ y) = y ^ x ^ y ( ^ (Bitwise xor) is associative )

y  ^ x ^ y = x ^ y ^ y ( ^ (Bitwise xor) is commutative )

x ^ y ^ y = x ^ 0 ( inverse of a number for bitwise xor is the number itself )

x ^ 0 = x (0 is the identity for XOR operation)

Hence if x < y, y ^ ((x ^ y) & -(x < y)) = x

If x >= y (2 Marks)

(x < y) = 0

-(x<y) = 0

(x ^ y)   & 0 = 0 (As any number AND 0 is 0)

y ^ (( x ^ y) & - (x < y) = y ^ 0

y ^ (( x ^ y) & - (x < y) = y (0 is the identity for XOR operation)

Hence if x >= y, y ^ ((x ^ y) & -(x < y)) = y
```

Therefore, this expression calculates the minimum of the two numbers

Grading Scheme

- 0 marks awarded for wrong option (False) with any justification
- 1 mark awarded for correct option (True) without correct justification.
- 1.5 marks awarded for correct option (True) and justifying it by correctly evaluating it for an example, incorrect evaluation of an example have been given 1.
- 3 marks awarded for correct option (True) and when only one case's justification is correct.
- 5 marks awarded for correct option (True) and correct justification by proving for both the cases.

Question 5 - 10 marks

Problem Description

Write a program to print the following pattern.

```
            1

        2       3

    4       5       6

7       8       9      10
```

Solution

```
#include<stdio.h>
int main() {
    int rows = 4, num = 1, space, i, j;
    for(i = 1; i <= rows; i++){
        for(space = 1; space <= rows - i; space++)
        printf("  ");
        for(j = i*(i-1)/2; j < i*(i+1)/2; j++){
            num = j + 1;
            printf("%d ", num);
        }
        printf("\n");
    }
}
```

Grading Scheme

- Header files: (**1 mark** - awarded if the syntax is strictly correct)
- Initialising variables using the right syntax: (**1 mark** - awarded if the syntax is strictly correct)
- Using the variables that were declared correctly: (**1 mark** - awarded with *some* leniency depending on the mistake)
- Using `printf` correctly for printing spaces:
  - **2 marks are** awarded if spaces are printed properly with correct ***generalised*** logic.
  - **1 mark is** awarded if just the syntax is correct.
- Using `for` or `while` loop:

    - **2 marks are** awarded if numbers are printed properly with correct ***generalised*** logic.
    - **1 mark is** awarded if just the syntax is correct
- Correct logic for ***generalised*** output - **2 marks**
- Correct final output (brute or general) - **1 mark**

Question 6 - 20 marks

Problem Description

What is output of the following programs with proper justification ?

i) 5 marks

```
#include<stdio.h>
int main()
{
    int x = 7, y, z;
    y = --x;
    z = x++;
    printf("%d %d %d\n", x, y, z);
    return 0;
}
```

Solution & Grading Scheme

Output: `7 6 6`

Justification:

```
y = --x; // y = 6, x = 6
z = x++; // z = 6, x = 7
```

`x = 7` (post-increment operator) **(2 marks)**
`y = 6` (pre-decrement operator) **(2 marks)**
`z = 6` (post-increment operator) **(1 mark)**

ii) 5 marks

```
#include<stdio.h>
int main()
{
    int i;
    for(i = 9 ;  i != 0 ; i--)
     printf("%d\n", i--);
    return 0;
}
```

Solution & Grading Scheme

Output: Infinite loop of odd numbers starting from `9, 7, 5, ....` **(2 marks)**
`i--` happens twice: once in the printf statement and once in the for loop.
The condition `i != 0` is never reached. So there is infinite loop. **(3 marks)**

iii) 5 marks

```
#include<stdio.h>
int main()
{
    int x = 4, y = 4, z = 4;
    if (x == y == z)
    {
        printf("YES");
    }

    else
    {
        printf("NO");
    }
    return 0;
}
```

Solution & Grading Scheme

Output: `NO`
(**2 marks**, only **1 mark** if justification is incorrect)

Expression gets evaluated from left to right `((x==y)==z)`
`x == y` is true, so it gets evaluated to `1`. (**1 mark**)

`1 == z` is false, so it gets evaluated to `0`. (**1 mark**)
Therefore “`NO`” is printed. **(1 mark)**

iv) 5 marks

```
#include<stdio.h>
int main()
{
    printf("%d", printf("Hello World"));
    return 0;
}
```

Solution & Grading Scheme

Output: `Hello World11`

Justification: printf outputs the text Hello World. Printf with `%d` returns the number of characters in the word `(11)`. Therefore, the output is `Hello World11`

2 marks for output and **3 marks** for justification