Question 1: What is the output for the following pseudo code

```
#include <stdio.h>

int fun(int n) {
    if(n == 0) // equality operator
        return 1;
    return n + fun(n - 2);
}

int main() {
    printf("%d", fun(4));
    return 0;
}
// Output: 6
```

Question 2: What is the output for the following pseudo code

```
#include <stdio.h>

int fun(int n) {
    if(n > 5) // relational >
        return n;
    return n + fun(n + 1);
}

int main() {
    printf("%d", fun(3));
    return 0;
}
// Output: 21
```

Question 3: What is the output for the following pseudo code

```
#include <stdio.h>

int fun(int n) {
    if(n < 2) // relational <
        return 1;
    return fun(n - 1) + fun(n - 2);
}

int main() {
    printf("%d", fun(5));
    return 0;
}
// Output: 8 (Fibonacci)
```

Question 4: What is the output for the following pseudo code

```
#include <stdio.h>

int fun(int n) {
    if(!n) // equivalent to n==0
        return 1;
    return n + fun(n - 1);
}

int main() {
    printf("%d", fun(4));
    return 0;
}
// Output: 10
```

Question 5: What is the output for the following pseudo code

```
#include <stdio.h>

int fun(int n) {
    if(n <= 0 && n >= -1) // using &&
        return 1;
    return n + fun(n - 2);
}

int main() {
    printf("%d", fun(5));
    return 0;
}
// Output: 9
```

Question 6: What is the output for the following pseudo code

```
#include <stdio.h>

int fun(int n) {
    if(n == 0 || n == 1) // logical OR
        return 1;
    return fun(n - 2) + fun(n - 1);
}

int main() {
    printf("%d", fun(4));
    return 0;
}
// Output: 4
```

Question 7: What is the output for the following pseudo code

```
#include <stdio.h>

int fun(int n) {
    if((n & 1) == 1) // bitwise AND to check odd
        return 1;
    return fun(n / 2) + n;
}

int main() {
    printf("%d", fun(6));
    return 0;
}
// Output: 12
```

Question 8: What is the output for the following pseudo code

```
#include <stdio.h>

int fun(int n) {
    if(n >= 5) // relational >=
        return n;
    return n + fun(n + 1);
}

int main() {
    printf("%d", fun(3));
    return 0;
}
// Output: 18
```

Question 9: What is the output for the following pseudo code

```
#include <stdio.h>

int fun(int n) {
    if(n <= 1) // relational <=
        return 2;
    return n + fun(n - 2);
}

int main() {
    printf("%d", fun(6));
    return 0;
}
```

```
// Output: 12
```

Question 10: What is the output for the following pseudo code

```
#include <stdio.h>

int fun(int n) {
    if(n % 2 == 0) // check even
        return 2;
    return fun(n - 1) + 3;
}

int main() {
    printf("%d", fun(5));
    return 0;
}
// Output: 8
```

Question 11: What is the output for the following pseudo code

```
#include <stdio.h>

int fun(int n) {
    if(!(n > 3)) // NOT operator
        return 1;
    return fun(n - 1) + 2;
}

int main() {
    printf("%d", fun(5));
    return 0;
}
// Output: 6
```

Question 12: What is the output for the following pseudo code

```
#include <stdio.h>

int fun(int n) {
    if((n & 1) == 0 && n < 3) // bitwise AND + &&
        return 1;
    return n + fun(n - 2);
}

int main() {
    printf("%d", fun(6));
    return 0;
}
// Output: 12
```

Question 13: What is the output for the following pseudo code

```
#include <stdio.h>

int fun(int n) {
    if(n != 0) // not equal
        return n + fun(n - 1);
    return 0;
}

int main() {
    printf("%d", fun(4));
    return 0;
}
// Output: 10
```

Question 14: What is the output for the following pseudo code

```
#include <stdio.h>

int fun(int n) {
    if(n % 2 == 0) // equality
        return 2;
    return n + fun(n - 1);
}

int main() {
    printf("%d", fun(5));
    return 0;
}
// Output: 8
```

Question 15: What is the output for the following pseudo code

```
#include <stdio.h>

int fun(int n) {
    if((n & 1) > 0) // check odd
        return 1;
    return n + fun(n / 2);
}

int main() {
    printf("%d", fun(6));
    return 0;
}
// Output: 12
```

Question 16: What is the output for the following pseudo code

```
#include <stdio.h>

int fun(int n) {
    if(n < 2 || n == 3)
        return 1;
    return fun(n - 1) + fun(n - 2);
}

int main() {
    printf("%d", fun(5));
    return 0;
}
// Output: 5
```

Question 17: What is the output for the following pseudo code

```
#include <stdio.h>

int fun(int n) {
    if(!n) // n==0
        return 1;
    return n + fun(n - 1);
}

int main() {
    printf("%d", fun(5));
    return 0;
}
// Output: 15
```

Question 18: What is the output for the following pseudo code

```
#include <stdio.h>

int fun(int n) {
    if((n & 1) == 0) // check even
        return 2;
    return fun(n - 1) + 3;
}

int main() {
    printf("%d", fun(5));
    return 0;
}
// Output: 8
```

Question 19:

```
#include <stdio.h>

int fun(int n) {
    if(n == 0 || n == 1)
        return 1;
    return n + fun(n - 2);
}

int main() {
    printf("%d", fun(4));
    return 0;
}
// Output: 6
```

Question 20:

```
#include <stdio.h>

int fun(int n) {
    if((n & 1) >= 1) // bitwise AND
        return 1;
    return n + fun(n - 2);
}

int main() {
    printf("%d", fun(6));
    return 0;
}
// Output: 12
```

Question: **Difference between break and continue with Example**

Break Statement:

Explanation:
The break statement is used to terminate the loop completely.
When break is executed, control comes out of the loop immediately, and no further iterations are executed.

Example (C language):

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

```
int main() {
    for(int i = 1; i <= 5; i++) {
        if(i == 3) {
            break;
        }
        printf("%d ", i);
    }
    return 0;
}
```

Output:

```
1 2
```

Explanation of Example:
The loop starts from 1
When i becomes 3, break executes
The loop stops completely
Numbers after 3 are not printed

Continue Statement:

Explanation:
The continue statement is used to skip the current iteration of the loop.
When continue is executed, control moves to the next iteration of the loop.

Example (C language):

```
#include <stdio.h>

int main() {
    for(int i = 1; i <= 5; i++) {
        if(i == 3) {
            continue;
        }
        printf("%d ", i);
    }
    return 0;
}
```

Output:

```
1 2 4 5
```

Explanation of Example:

The loop runs from 1 to 5
When i becomes 3, continue executes
Printing of 3 is skipped
The loop continues with the next iteration