Question 1. [25 marks for whole question] We consider the problem of generating binary strings of length \( n \) in which a sub-string “11” is forbidden. Such strings are called 11-free strings. The following are examples of 11-free strings for \( n=1, 2, 3, 4 \) given in lexicographic order.

<table>
<thead>
<tr>
<th>( n=1 )</th>
<th>( n=2 )</th>
<th>( n=3 )</th>
<th>( n=4 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0 0</td>
<td>0 0 0</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>1</td>
<td>0 1</td>
<td>0 0 1</td>
<td>0 0 0 1</td>
</tr>
<tr>
<td></td>
<td>1 0</td>
<td>0 1 0</td>
<td>0 0 1 0</td>
</tr>
<tr>
<td></td>
<td>1 0 0</td>
<td>0 1 0 0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 0 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 0 0 0</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 0 0 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 0 1 0</td>
<td></td>
</tr>
</tbody>
</table>

(a) [2 marks] List up 11-free strings of length 5 in lexicographic order. How many strings are there?

Let \( F(n) \) be the set of such strings given in lexicographic order. Then we have the following recurrence for \( F(n) \).

\[
F(0) = \text{empty} \quad (\text{empty means an empty string, that is, the string of length 0})
\]

\[
F(1) = \{0, 1\}
\]

\[
F(n) = 0F(n-1) \cup 10F(n-2), \quad (n \geq 2)
\]

Here the concatenation of a string and a set is order preserving, e.g.,

\( 0F(2) = \{000,001,010\} \) etc. Let the Fibonacci sequence of integers be defined by

\[
f(0)=0, \quad f(1)=1
\]

\[
f(n)=f(n-1) + f(n-2), \quad (n \geq 2)
\]

Since the two sets of the right-hand side of the recurrence for \( F(n) \) are mutually disjoint, we see that the size \( |F(n)| \) is given by the \( (n+2) \)th Fibonacci number \( f(n+2) \). Confirm this for \( F(5) \).

Recall that the set of unrestricted binary strings of length \( n \), \( B(n) \), was given by the recurrence

\[
B(0) = \text{empty}
\]

\[
B(n) = 0B(n-1) \cup 1B(n-1), \quad (n \geq 1)
\]

Based on this recurrence, we had a recursive algorithm as shown below. The procedure \( \text{output}(a) \) is to output \( a[n], ..., a[1] \) in this order to output the string contained in \( a \).
Algorithm 1.

procedure binary(n);
begin
  if n>0 then begin
    a[n]:=0; binary(n-1);
    a[n]:=1; binary(n-1)
  end else output(a)
end;

begin {main program}
  binary(n)
end.

(b) [3 marks] Write a recursive algorithm for generating $F(n)$. A pseudo code is acceptable.

Recall that an iterative algorithm for $B(n)$ was given by the following.

**Algorithm 2**

for i:=1 to n do a[i]:=0;
repeat
  output(a);
  i:=1;
  while a[i+1]=1 do
    a[i+1]:=0;
    i:=i+1;
  end;
  a[i+1]:=1
until i=n.

(c) [5 marks] Write an iterative algorithm for generating $F(n)$. A pseudo code is acceptable.

Recall that the Gray code for binary strings was based on the recurrence

$$G(0) = \text{empty}$$
$$G(n) = 0G(n-1) \cup 1G'(n-1), \ (n \geq 1)$$

$$G'(0) = \text{empty}$$
$$G'(n) = 1G(n-1) \cup 0G'(n-1), \ (n \geq 1).$$

where $G'(n)$ is the Gray code of length $n$ arranged in reverse order. Note that we started from $G(1)$ in the notes. Based on this recurrence, we had the following recursive algorithm for the Gray code for binary strings.

**Algorithm 3. Recursive Gray code**
procedure Gray1(n);
begin
    if n>0 then begin
        a[n]:=0; Gray1(n-1);
        a[n]:=1; Gray2(n-1)
    end else output(a)
end;

procedure Gray2(n);
begin
    if n>0 then begin
        a[n]:=1; Gray1(n-1);
        a[n]:=0; Gray2(n-1)
    end else output(a)
end;

begin {main program}
    Gray1(n)
end.

Following this example, we can establish the following recurrence for the Gray code of 11-free strings in the following.

\[ H(0) = \text{empty}, \quad H(1) = \{0, 1\} \]
\[ H(n) = 0H(n-1) \cup 10H'(n-2), \quad (n \geq 2) \]

\[ H'(0) = \text{empty}, \quad H'(1) = \{1, 0\} \]
\[ H'(n) = 10H(n-2) \cup 0H'(n-1), \quad (n \geq 2) \]

(d) [2 marks] Base on the above recurrence, we have some examples below

<table>
<thead>
<tr>
<th>n=1</th>
<th>n=2</th>
<th>n=3</th>
<th>n=4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0 0</td>
<td>0 0 0</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>1</td>
<td>0 1</td>
<td>0 0 1</td>
<td>0 0 1</td>
</tr>
<tr>
<td></td>
<td>1 0</td>
<td>0 1 0</td>
<td>0 0 1</td>
</tr>
<tr>
<td></td>
<td>1 0 1</td>
<td>0 1 0 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 0 0</td>
<td>0 1 0 0</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 0 1 0</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 0 0 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 0 0 0</td>
<td></td>
</tr>
</tbody>
</table>

List up H(5).

(e) [4 marks] Write a recursive algorithm for H(n). A pseudo code is acceptable.

(f) [4 marks] Prove by induction that the first string of H(n) and the last string of H'(n) are 00...0 (n 0’s), and that the last string of of H(n) and the first string of H'(n) are 100...0 (1 followed by n-1 0’s). From this observation, prove that we have at most 3 changes from a string to the next in H(n).
Based on the previous observation, we wish to design an iterative algorithm for $H(n)$. Recall that the iterative algorithm for the binary reflected Gray code was given as follows:

```plaintext
program ex(input,output);
var i,n:integer;
a,up:array[0..100] of integer;
procedure out;
var i:integer;
begin
  for i:=1 to n do write(a[i]:2);
  writeln
end;
begin
  readln(n);
  for i:=1 to n do a[i]:=0;
  for i:=0 to n do up[i]:=i;
  repeat
    out;
    i:=up[n];
    up[n]:=n;
    a[i]:=1 - a[i];
    up[i]:=up[i-1]; /* The value of up propagates */
    up[i-1]:=i-1;
  until i=0;
end.
```

Array "up" is to guide us to go up the tree from the leaf level as illustrated in the following figure.

```plaintext
level
0 1 2 3 4
S
0 0 0 0
0 1 0 0 1
B 1 1 0 0 1
0 1 0 0 1
C 1 0 1 1 0
1 1 0 0 1
0 1 0 0 1
0 1 1 0 0
1 1 0 0 1
0 1 1 1 1
1 1 1 1 1
0 1 1 0 1
1 1 0 1 1
0 1 0 1 1
0 1 0 0 1
1 1 1 1 1
0 1 1 1 0
1 1 1 0 0
0 1 0 0 1
1 1 0 0 1
0 1 0 0 1
```

Array “up” is to guide us to go up the tree from the leaf level as illustrated in the following figure.
We generate bit strings by traversing the above tree starting from S. The general move of “up”, “cross”, and “down” is demonstrated by the move (...,A,B,C,D, ...). The move A -> B is guided by the array “up”. At position A, we have up[4]=2.

Draw a similar tree for H(5), and complete the following algorithm for H(n). Specifically you fill the blank parts shown by underlines. Note that we need more maintenance work for “up” as a node with label 1 has only one child and we need to propagate the value of “up” towards that child.

```pascal
program ex(input,output);
var i,n:integer;
    a,up:array[0..100] of integer;
procedure out;
var j:integer;
begin
    for j:=1 to n do write(a[j]:2);
    writeln
end;
begin
    readln(n);
    for i:=1 to n do a[i]:=0;
    for i:=0 to n do up[i]:=i;
    out;
repeat
    i:=up[n];
    up[n]:=n;
    a[i+1]:=1-a[i];
    a[i+2]:=1-a[i+2];
    if i>0 then out;
    up[i]:=up[i-1];
    up[i-1]:=i-1;
    if a[i]=1 then begin
        up[i+1]:=up[i]; up[i+3]:=______(B)
    end
    else up[i+2]:=up[i+1];
until i=0;
end.
```

Give your answers referring to labels (A) and (B).

Question 2 [25 marks for the whole question]. We consider a hybrid sorting network consisting of odd-even and bitonic merging networks, which sort n input numbers in increasing order, where n is a power of 2. We design this network with two identical bitonic sorting networks with n/2 inputs followed by an odd-even merging network with n inputs. The following is an example with with 8 inputs. Stage i consists of the networks for merging a bitonic sequence of length 2^i, for i <= log_2 n - 1 The (+)-element sorts two elements in increasing order, and the (-)-element in decreasing order.
Based on this example, draw a hybrid sorting network with 16 inputs, and trace it with inputs (2, 4, 6, 8, 10, 12, 14, 16, 1, 3, 5, 6, 7, 9, 11, 13, 15).