Questions

2. a) Define a C++ function that returns the k^{th} term of a recurrence relation. Specifically the file containing your definition must `#include` the file `Q2.h` on Blackboard, and so provide a definition matching the declaration

```
int recurrence(int p, int q, int k, int u0);
```

The function should return the value of u_k, defined by the recurrence

$$u_{n+1} = pu_n + q \qquad \text{for } n \geq 0 \text{ with } u_0 \text{ as supplied.}$$

Demonstrate your function, showing the output corresponding to:

See the Introduction for your values of a and b.

- `p` $= a$, `q` $= b$, `k` $= 0$, `u0` $= 3$
- `p` $= 10 + a$, `q` $= 10 + b$, `k` $= 5$, `u0` $= 2$

b) In the same file, define a second C++ function that uses a `vector` to return $u_0, \ldots, u_k$ for the recurrence below. Specifically, your function should match declaration

```
void LFSR(int k, vector<int> &u);
```

where `u` is a `vector` passed by reference, initially containing the values u_0, u_1, u_2, u_3 in positions 0 to 3. When the function is called, if $k > 3$, the extra values of u_i for $i = 4, \ldots, k$ should be placed in positions $i = 4, \ldots, k$. They should be calculated using

$$u_{n+4} = u_{n+1} + u_n \mod 2 \text{ for } n \geq 0$$

Demonstrate your function, showing the output produced for the sequence starting with $u_i = a_i$ for $i = 0, 1, 2, 3$. Use your function to calculate the values of u_i for $i = 0 \ldots 49$. State how long the sequence continues before starting to repeat. (*Sequences such as these have many surprising properties and are generated by hardware called Linear Feedback Shift Registers, and widely used in cryptography, error-correcting codes, and radar.*)

See the Introduction for your values of a_i.

6. a) Define a C++ function that returns the k^{th} term of a recurrence relation. Specifically the file containing your definition must `#include` the file `Q6.h` on Blackboard, and so provide a definition matching the declaration

```
int recurrence(int p, int q, int k, int u0, int u1);
```

and should return the value of u_k, defined by the recurrence

$$u_{n+2} = pu_{n+1} + qu_n \qquad \text{for } n \geq 0 \text{ with } u_0, u_1 \text{ as supplied.}$$

Demonstrate your function, showing the output corresponding to:

See the Introduction for your values of a and b.

- $\texttt{p} = a$, $\texttt{q} = b$, $\texttt{k} = 0$, $\texttt{u0} = 3$, $\texttt{u1} = 5$
- $\texttt{p} = 10 + a$, $\texttt{q} = 10 + b$, $\texttt{k} = 5$, $\texttt{u0} = 1$, $\texttt{u1} = 2$

b) In the same file, define a second C++ function that uses a `vector` to return $u_0, \ldots, u_k$ for the recurrence below. Specifically, your function should match declaration

```
void LFSR(int k, vector<int>& u);
```

where `u` is a `vector` passed by reference, initially containing the values u_0, u_1, u_2, u_3, u_4 in positions 0 to 4. When the function is called, if $k > 4$, the extra values of u_i for $i = 5, \ldots, k$ should be placed in positions $i = 5, \ldots, k$. They should be calculated using

$$u_{n+5} = u_{n+2} + u_n \quad \text{mod } 2 \text{ for } n \geq 0$$

Demonstrate your function, showing the output produced for the sequence starting with $u_i = a_i$ for $i = 0, 1, 2, 3$ and $u_4 = 1$. Use your function to calculate the values of u_i for $i = 0 \ldots 49$. (This should produce a sequence that repeats every 31 elements). Take the output of length 31 and cyclically shift it by 1 place, then compare this with the unshifted sequence. In how many places does it agree and how many does it disagree?

See the Introduction for your values of a_i.

8. Submit an improved version of the program you submitted in question 4. Specifically the file containing your definitions must `#include` the file `Q8.h` on Blackboard, your definition of `shortest` should now match the modified declaration

   - `int shortest_path(int vertex1, int vertex2, vector<int> &u)`

   On return, the vector `u` should contain the vertices that make up the shortest path, with first vertex `v1` in `u[0]`, and last vertex `v2` in `u[`$m-1$`]`, where m is the number of vertices in the path. (Note: the edge weights for this question are now integers, to avoid any problems comparing `floats` for equality).

   Demonstrate your class by writing a program using a `Graph` object to read in `Q8_graphdata.txt` on Blackboard, `print` its contents, calculate and print the shortest distance between vertices 0 and 3 and the corresponding path. There are 10 marks for the improved version of your original program, and 10 marks for new bits of the implementation of `shortest_path`.

10. a) Draw a finite state automaton (FSA) accepting all binary strings containing an even number of 1s. (3 marks)

    b) Draw an FSA accepting all binary strings containing exactly two 1s.

    c) Draw an FSA accepting all binary strings containing a number of 1s that is a multiple of 2 or 3.

    d) Define a C++ class `FSA` that represents finite state automata. Specifically the file containing your definitions must `#include` the file `Q10.h` on Blackboard. You need to provide a definition for the method

       - `bool accepts(vector<int> inputs)`

       Demonstrate your class by writing a program to implement the FSA in part a) above. Produce output to demonstrate if your program accepts input 101101.