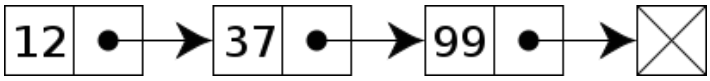
About this test:

- The test is **not** eliminatory
- It is a preparation for the interview
- You can use any reference material or even learn from other people
- Some concepts discussed here may be used during the technical interview

Contents

1. Algorithms and Data Structures

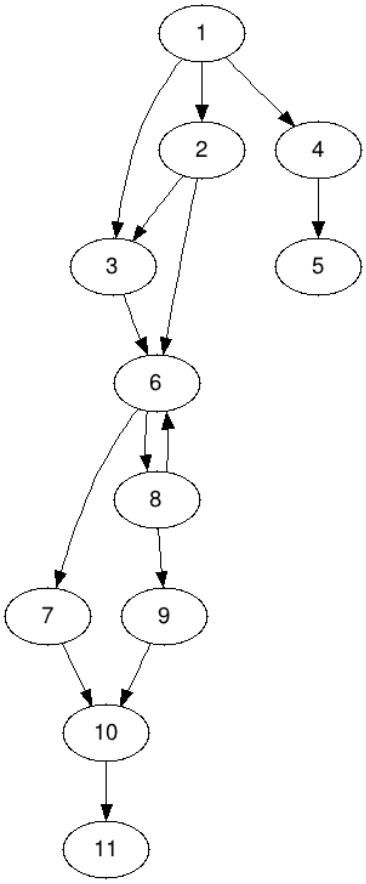
1.1. You were given a singly linked list of integers:



You need to return a list with the elements in the reverse order. Describe an algorithm, for the generic case of a list containing N elements that would do such work in a linear time complexity.

1.2. You are developing a function to find an element in an array. The number of elements of the array is unknown and the elements are sorted. If you access a position out of the bounds of the array, an exception is thrown. Describe an efficient algorithm to search an element.

1.3. Consider the following graph:



a) Starting on node 1, and always visiting nodes from left to the right:
   i. Write the numbers corresponding to the nodes in the order that they would be visited if a BFS was used
   ii. Now do the same using a DFS
b) Describe an algorithm that could be used to find a path between two nodes of this graph. Give a high-level explanation of it and justify why you chose it
c) Do the same for an algorithm which finds the shortest path
d) Do the same for an algorithm which finds the shortest path going through a specific node

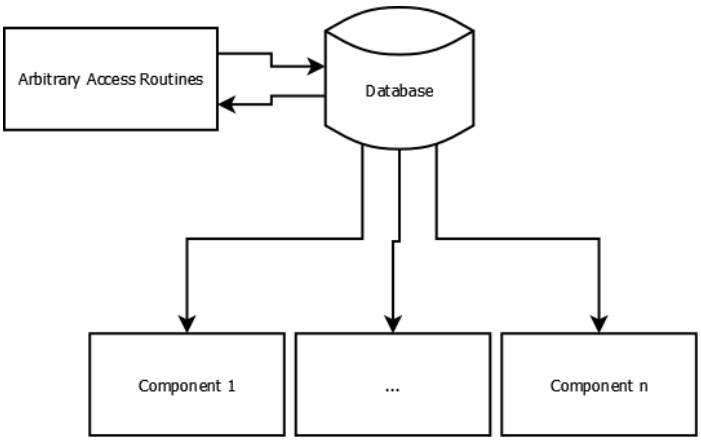
2. Software Design

2.1. The following class, `SimpleJobDispatcher`, implements part of a simple job dispatcher to a computer farm:

```
class SimpleJobDispatcher {
    public static void main (String [] args) throws IOException {
        JobDispatcher jobDisp = new JobDispatcher ();
        for (; ; ;) { // just an illustrative loop
            Job job = jobDisp.dispatch();
            DoSomeProcessing(job);
        }
    }
}
```

After some time, you find that the `SimpleJobDispatcher` has poor performance because `jobDisp.dispatch()` frequently asks the Farm Manager to create slots for new jobs. What steps could you take to improve `SimpleJobDispatcher`'s performance?

2.2. Suppose you need to develop a software that contains a database manipulated by an arbitrary set of routines. This database is also read by many other components of the software to do component-specific calculations, as the figure shows:



Based on the architecture shown above:

a) Describe a way that the components could be aware of any changes in the database without querying it periodically.

b) How would you improve solution you proposed in a) to handle concurrent access to the database?

2.3. You have a program written in C language that allocates a huge array of integers. You would like to initialize all the entries to **0**, but the length of the array is too big, and you have to avoid the linear time complexity of initialization.
Design a deterministic scheme by which reads and writes to the array can be made in constant time complexity and assures that read operations always get initialized values. Assume memory consumption is not a constraint.

3. C language

3.1. Locate bugs in the following program:

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

#define BUFF_SIZE 100
char *function1(const char *str) {
    char buff[BUFF_SIZE];
    int i;
    size_t strSize = strlen(str);
    for (i = 0 ;
         i < strSize && i < BUFF_SIZE ;
         i++) {
        char c = str[i];
        if (c >= 'a' && c <= 'z') {
            c = (c - 'a') + 'A';
        }
        buff[i] = c;
    }
    return buff;
}

int main() {
    char *msg = "Hello world!";
    char *ptr;
    printf("String Fun!\n");
    printf("Original msg is %s\n",msg);
    ptr = function1(msg);
    printf("ptr is %s\n",ptr);
    printf("msg is %s\n",msg);
    return 0;
}
```

3.2. Locate bugs in the following program:

```
typedef struct List {
    int *data;
    struct List *next;
} listT;

void cleanList(listT *myListP);
void doSth();
int condition();

void main()
{
    listT *myListP = NULL;
    if (condition()) {
        cleanList(myListP);
    }
    if (myListP != NULL) {
        doSth();
    }
}

void cleanList(listT *myListP)
{
    while(myListP != NULL) {
        free(myListP);
        myListP = myListP->next;
    }
}
```

3.3. Note the performance issues with the following code

```
#include <stdlib.h>
#include <string.h>
char *my_strchr(const char *str, char C)
{
    int loop = 0;
    while (str[loop] != '\0')
    {
        if (str[loop] == C)   return &str[loop];
        loop++;
    }
    return NULL;
}

int some_function(const char *string)
{
    int loop;
    int numA = 0;

    for (loop = 0;  loop< strlen(string); loop++)
    {
        if ((char*) NULL != my_strchr(string[loop], 'A'))   numA++;
    }
    return numA;
}
```

3.4. Rewrite the code in the question above more efficiently.

3.5. Which of the following gives the memory address of the first element in array?

(a) array [0]

(b) array[1]

(c) array(2)

(d) array

3.6. On freeing a dynamic memory, if the pointer value is not modified, then the pointer points to.

(a) NULL

(b) Other dynamically allocated memory

(c) The same deallocated memory location

(d) It points back to location it was initialized with

3.7. What is the type of variable ‘z’ in following code?

```
#include <stdio.h>
int main()
{
    int z;
}
```

(a) static variable

(b) automatic variable

(c) register variable

(d) global variable.

4. C++ and STL

4.1. In STL, which are the main differences between a vector and a list?

4.2. Suppose you need to count a list of the unique words that are read from an `istream`.

a) Write the prototype for the method you are going to create.
b) Which STL structure would you use to store the words?
c) Write the code to implement what is proposed in this problem.
d) If you were not using the STL container mentioned in b), would you propose another alternative STL container or even develop another container? Which? (There is no need to write program, just an explanation)

4.3. Consider the partial definition of the Student class below and the `FindCourseName` method:

```
class Student {
       // many attributes and methods here plus:
       char* name; // Student name
       char* courseName; // Student course name
       std::string toString() { return std::string(name); }
}

std::string FindCourseName ( std::list< Student > stu, string name )
{
    for ( std::list< Student >::iterator it = stu.begin();
       it != stu.end();
       it++ )
       {
           if ( (*it).toString() == name )
           {
                      return it->courseName;
           }
       }
    return "";
}
```

a) How many unnecessary object creations/copies are done in the FindCourseName method?
b) How would you reduce this number of copies?
c) How would you optimize the code above?

4.4. Remove the errors and optimize the code below.

```
typedef std::vector< std::string > StringVector;
bool myMethod ( const StringVector input, StringVector& output )
{
    StringVector::iterator first = find ( input.begin(), input.end(), "Fabiano" );
    StringVector::iterator last = find ( input.begin(), input.end(), "Vector" );
    *last = "Victor";
    for ( StringVector::iterator it = first;
        it != last;
        ++it )
    {
        std::cout << *it << std::endl;
    }

    input.insert( --(input.end()), "XXX" );
    if ( first != input.end() ) std::cout << *first << std::endl;
    for ( StringVector::iterator it = input.begin();
        it != input.end();
        ++it )
    {
        output.push_back( *it );
    }
}
```

4.5. Consider the following C++ program:

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

class MyClassOne {
  public:
    int calculate() { return 1; }
};

class MyClassTwo : public MyClassOne {
  public:
    virtual int calculate() { return 2; }
};

class MyClassThree : public MyClassTwo {
  public:
    int calculate() { return 3; }
};

int main() {
    int result = 0;
    MyClassOne *objs[3];
           objs[0] = new MyClassOne();
           objs[1] = new MyClassTwo();
           objs[2] = new MyClassThree();

           for(int i = 0; i < 3; ++i) {
               cout << objs[i]->calculate() << endl;
           }

           return 0;
       }
```

What is going to be printed to the standard output?

5. C++11

5.1. Describe what happens in the following code (suppose move constructors and operators are defined in BigInt class).

```
void swap(BigInt& a, BigInt& b) {
    BigInt t(move(a));
    a = move (b);
    b = move (t);
}
```

5.2. Implement move constructor and operator for BigInt class. Consider the following initial description for the class:

```
class BigInt {
public:
    BigInt(unsigned len);
    BigInt(const BigInt& bi);
    BigInt& operator=(const BigInt& bi);
    ~BigInt();
private:
    unsigned len_;
    uint8_t* mag_;
};
BigInt::BigInt(unsigned len)
        : len_(len)
        , mag_(len ? new uint8_t[len] : nullptr)
{}
BigInt::BigInt(const BigInt& bi)
        : len_(bi.len_)
        , mag_(len_ ? new uint8_t[bi.len_] : nullptr)
{
    std::copy(bi.mag_, bi.mag_ + len_, mag_);
}
BigInt&
BigInt::operator=(const BigInt& bi)
{
   if (&bi == this) return *this;
   len_ = bi.len_;
   delete [] mag_;
   mag_ = len_ ? new uint8_t[bi.len_] : nullptr;
   std::copy(bi.mag_, bi.mag_ + len_, mag_);
   return *this;
}
BigInt::~BigInt() {
   delete [] mag_;
}
```

6. Hardware and Simulation

6.1. Draw the circuit that would be generated from the Verilog/VHDL description right below. Both descriptions are equivalent.

Verilog

```
module test (clock, reset, sel, result);
  input clock, reset, sel;
  output result;
  reg [1:0] q;
  wire [1:0] i;

  assign i = {q[0],sel};
  always @(posedge clock)
      if (reset) q = 2'b0;
      else q = i;

  assign result = (q == i) ? 1'b1 : 1'b0;

endmodule
```

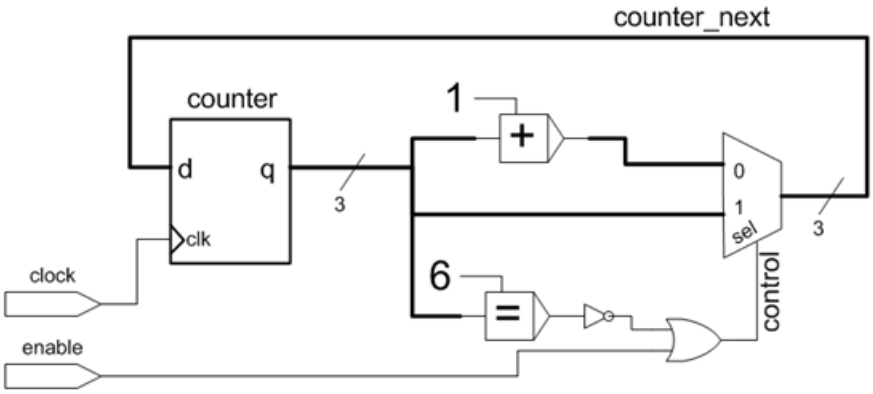
VHDL

```
library IEEE;
use IEEE.std_logic_1164.all;

entity test is
  port (
      clock,reset,sel: in std_logic;
      result: out std_logic
  );
end entity;

architecture rtl of test is
  signal q,i: std_logic_vector(1 downto 0);
begin
  i <= q(0) & sel;
  process (clock, reset)
  begin
      if (rising_edge(clock)) then
      if (reset = '1') then
      q <= "00";
      else
      q <= i;
      end if;
      end if;
  end process;
  result <= '1' when (q = i) else '0';
end architecture;
```

6.2. Consider the circuit below:

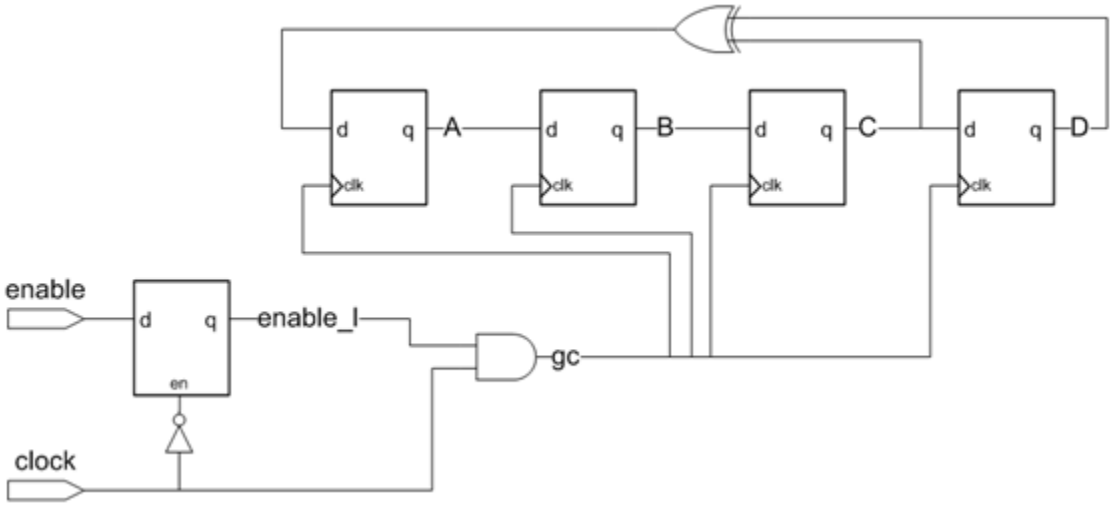


Complete the state transition diagram below for the register “counter”. Consider that the finite-state machine makes a transition when “clock” has a positive edge and that every signal can only take the values 0 or 1 (no X values).

Tip: Don’t assume anything about the circuit behavior based on signal names.

0	1	2	3
7	6	5	4

6.3. Given the circuit below, complete the waveform:



Tip: The green-filled cycles are considered as an undefined value X, such as a memory element that has not been initialized or reseted.