2008 Paper 3 Question 3

Programming in C and C++

A hardware engineer stores a FIFO queue of bits in an `int` on a platform with 32-bit `int`s and 8-bit `char`s using the following C++ class:

```
class BitQueue {
  int valid_bits; //the number of valid bits held in queue
  int queue;      //least significant bit is most recent bit added
 public:
  BitQueue(): valid_bits(0),queue(0) {}
  void push(int val, int bsize);
  int pop(int bsize);
  int size();
};
```

(*a*) Write an implementation of `BitQueue::size`, which should return the number of bits currently held in `queue`. [1 mark]

(*b*) Write an implementation of `BitQueue::push`, which places the `bsize` least significant bits from `val` onto `queue` and updates `valid_bits`. An exception should be thrown in cases where data would otherwise be lost. [5 marks]

(*c*) Write an implementation of `BitQueue::pop`, which takes `bsize` bits from `queue`, provides them as the `bsize` least significant bits in the return value, and updates `valid_bits`. An exception should be thrown when any requested data is unavailable. [4 marks]

(*d*) The hardware engineer has built a communication device together with a C++ library function `send` to transmit data with the following declaration:

```
void send(char);
```

Use the `BitQueue` class to write a C++ definition for:

```
void sendmsg(const char* msg);
```

Each of the characters in `msg` should be encoded, in index order, using the following binary codes: `'a'`=`0`, `'b'`=`10`, `'c'`=`1100`, and `'d'`=`1101`. All other characters should be ignored. Successive binary codes should be bit-packed together and the code `111` should be used to denote the end of the message. Chunks of 8-bits should be sent using the `send` function and any remaining bits at the end of a message should be padded with zeros. For example, executing `sendmsg("abcd")` should call the `send` function twice, with the binary values `01011001` followed by `10111100`. [10 marks]