SOLUTIONS 2

Answers to the BONUS questions are available by request via email.


   ```prolog
   flip(0,1). % part i
   flip(1,0).

   all0([]). all0([H|T]) :-
       H = 0,
       all0(T).

   has0([H|T]) :- % part ii)
       ( H=0 ->
           true
       ;
           has0(T)
       ).

   decreasing([]). decreasing([H|T]) :-
       ( H=1 ->
           decreasing(T)
       ;
           all0(T)
       ).

   continue([H|T]) :-
       ( H=1 ->
           true
       ;
           has0(T) % part ii)
       ).

   next([H|T], X) :-
       continue([H|T]),
       ( decreasing(T) ->
           flip(H,I), % part iii)
           append(T,[I],X)
       ;
           append(D,[0,1|C],T),
           decreasing(D),
           append(D,[0,1,H|C], X)
       ).

   cool(A,C) :-
       next(A,C).
   cool(A,C) :-
       next(A,B),
       cool(B,C).
   ```

1 ?- consult(question1).
% question1 compiled 0.00 sec, 0 bytes

Yes

2 ?- cool([1,1,1,1],X).

| X = [1, 1, 1, 0] |
| X = [1, 1, 0, 0] |
| X = [1, 0, 0, 0] |
| X = [0, 0, 0, 0] |
| X = [0, 0, 0, 1] |
| X = [0, 0, 1, 0] |
| X = [0, 1, 0, 0] |
| X = [0, 1, 0, 1] |
| X = [0, 1, 1, 0] |
| X = [0, 1, 1, 1] |
| X = [1, 0, 1, 0] |
| X = [1, 0, 1, 1] |
| X = [1, 1, 0, 0] |
| X = [1, 1, 0, 1] |
| X = [1, 1, 1, 0] |
| X = [1, 1, 1, 1] |

No
2. a) See Figure 1.

![Diagram of a BDD](image)

**Figure 1.** The unreduced independent set BDD $B$ for your graph $G$.

b) The BDD represented in a file and the result of using `write_bdd`.

```prolog
get_bdd(b,
  bdd(1,
    bdd(2,
      bdd(3,
        bdd(4,
          bdd(leaf,t,x),
          bdd(leaf,t,x)
        ),
        bdd(4,
          bdd(leaf,t,x),
          bdd(leaf,f,x)
        )
      ),
      bdd(3,
        bdd(4,
          bdd(leaf,t,x),
          bdd(leaf,t,x)
        ),
        bdd(4,
          bdd(leaf,t,x),
          bdd(leaf,f,x)
        )
      )
    ),
    bdd(2,
      bdd(3,
        bdd(4,
          bdd(leaf,t,x),
          bdd(leaf,f,x)
        ),
        bdd(4,
          bdd(leaf,t,x),
          bdd(leaf,f,x)
        )
      )
    )
  ),
  bdd(2,
    bdd(3,
      bdd(4,
        bdd(leaf,t,x),
        bdd(leaf,f,x)
      ),
      bdd(4,
        bdd(leaf,t,x),
        bdd(leaf,f,x)
      )
    )
  )
).
```

1 ?- consult(bdd/bdd), consult(bdd/bdd-t), consult(bdd/bddwrite).
Yes
2 ?- consult(question2).
Yes
3 ?- get_bdd(b,B), write_bdd(B).

---

1. v1
2. v2
3. v3
4. v4
5. t
6. v4
7. f
8. v3
9. v2
10. f
11. v3

---

c) See Figure 2.
d) Output below.

```prolog
4 ?- get_bdd(b,B), reduce(B,R), write_bdd(R).
   [1] v1
   [2]   v2
   [3]     v3
   [4]       t
   [5]         v4
   [6]           f

R = bdd(1, bdd(2, bdd(3, bdd(leaf, t, x), bdd(4, bdd(leaf, t, x), bdd(leaf, f, x)))), bdd(4, bdd(leaf, t, x), bdd(leaf, f, x))), bdd(4, bdd(leaf, t, x), bdd(leaf, f, x)))
```

Figure 2. The reduced BDD $R$ from the unreduced BDD $B$. 

Reduction Type 1

Reduction Type 2

Reduction Type 3

Reduction Type 2

Reduction Type 2

Reduction Type 2
e) See Figure 3 for the reduced BDD. The two reduced BDDs are the same. They are the same because they have the same proposition order (1,2,3,4) and represent the same formula. Any two reduced BDDs representing the same formula and having the same proposition order are identical.

![Figure 3. The result of reducing BDD B by using Prolog.](image)

f) The number of satisfying assignments represents the number of distinct independent sets. The value of numSAT is computed below.

```prolog
7 ?- get_bdd(b,B), numSAT(B,2,N).
```

N = 9

g) The correct number of satisfying assignments is $2^{n_1-n_1-1}s_1 + 2^{n_2-n_2-1}s_2$ because there are $n - n_1 - 1$ skipped predicates on one branch and $n - n_2 - 1$ skipped predicates on the other branch. Each skipped predicate can be set to two values (True for False) in each satisfying assignment.