Homework of Data Structures

Chapter 1 Introduction

Exercises 1.1 The Game of Life

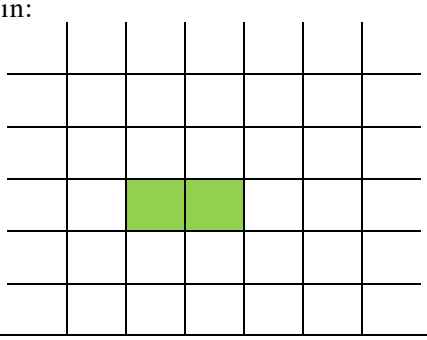
The Game of Life, also known simply as Life, is a cellular automaton devised by the British mathematician John Horton Conway in 1970. In this game, life is really a simulation, not a game with players. The universe of the Game of Life is an infinite, two-dimensional orthogonal grid of square cells, each of which is in one of two possible states, alive or dead (occupied cells are called alive; unoccupied cells are celled dead.), which cells are alive changes from generation to generation according the number of neighboring cells that are alive, as follows:

1) The neighbors of a given cell are the eight cells that touch it horizontally, vertically, or diagonally adjacent.
2) Any live cell with fewer than two live neighbors dies of loneliness on to the next generation.
3) Any live cell with two or three live neighbors lives on to the next generation.
4) Any live cell with four or more live neighbors dies of overcrowding.
5) Any dead cell with exactly three live neighbors becomes a live cell, as if by reproduction. All other dead cells remain dead in the next generation.
6) All births and deaths take place at exactly the same time, so that dying cells can help to give birth to another, but cannot prevent the death of others by reducing overcrowding, nor can cells being born either preserve or kill cells living in the previous generation.

【Example】

- **Dying:** by rule 2 both the living cells will die in the coming generation, and rule 5 shows that no cells will become alive, so the community dies out.

Begin:



Mark living neighbor:

0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	1	2	2	1	0	0
0	1	**1**	**1**	1	0	0
0	1	2	2	1	0	0
0	0	0	0	0	0	0

Mark change:

0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	1	2	2	1	0	0
0	1	**1**	**1**	1	0	0
0	1	2	2	1	0	0
0	0	0	0	0	0	0

The 1st generation:

Mark change:

0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

The 2st generation:

Figure 1(a) Dying community

- **Stability:** the community below has the neighbor counts as shown. Each of the living cells has a neighbor count of three, and hence remains alive, but the dead cells all have neighbor counts of two or less, and hence none of them becomes alive.

Begin:

Mark living neighbor:

0	1	2	2	1	0	0
0	2	3	3	2	0	0
0	2	3	3	2	0	0
0	1	2	2	1	0	0
0	0	0	0	0	0	0

Mark change:

0	1	2	2	1	0	0
0	2	3	3	2	0	0
0	2	3	3	2	0	0
0	1	2	2	1	0	0
0	0	0	0	0	0	0

The 1st generation:

The 2nd generation:

The 3rd generation:

Figure 1(b) Stability community

- **Alternation:** the two communities below continue to alternate from generation to generation, as indicated by the neighbor counts shown.

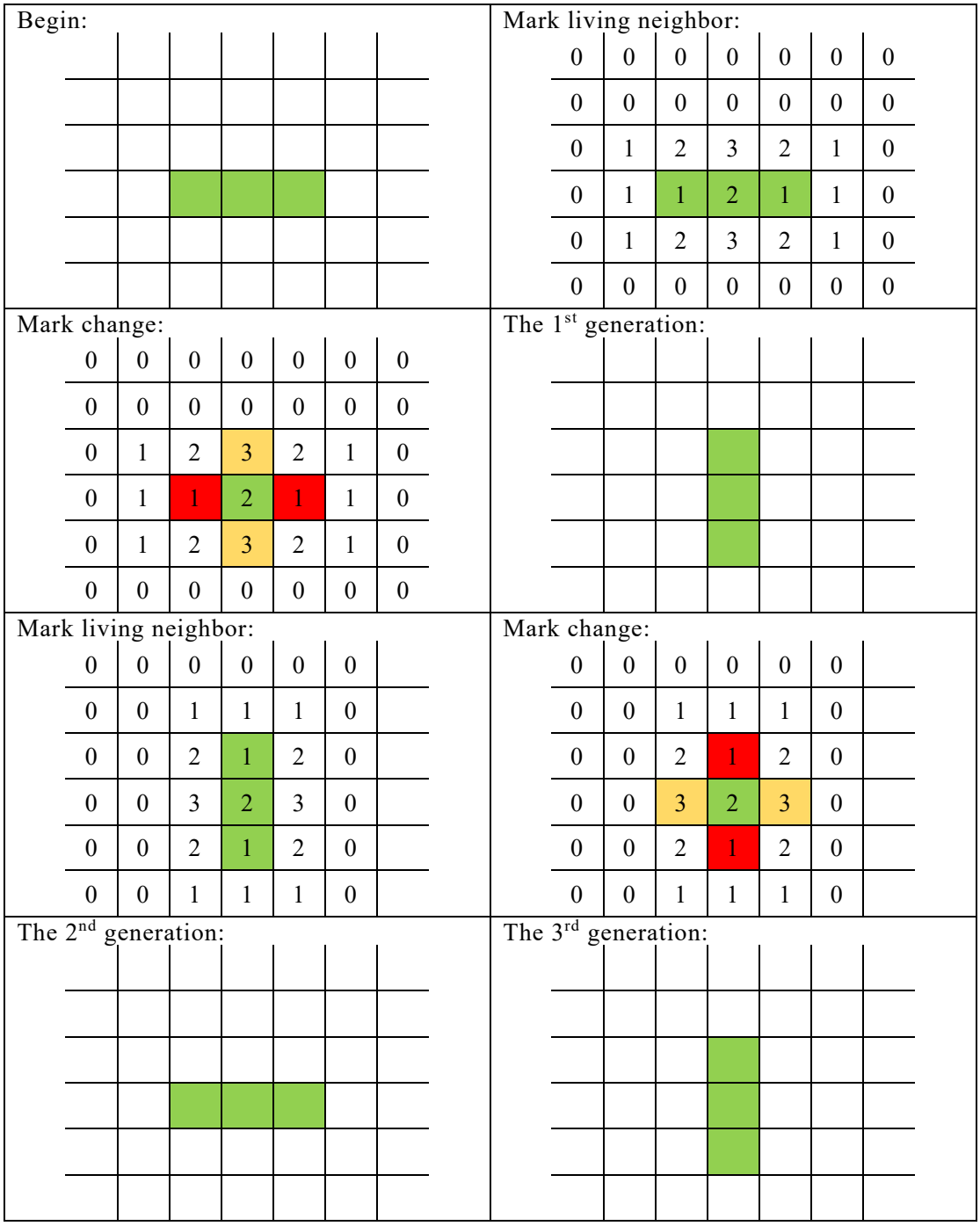


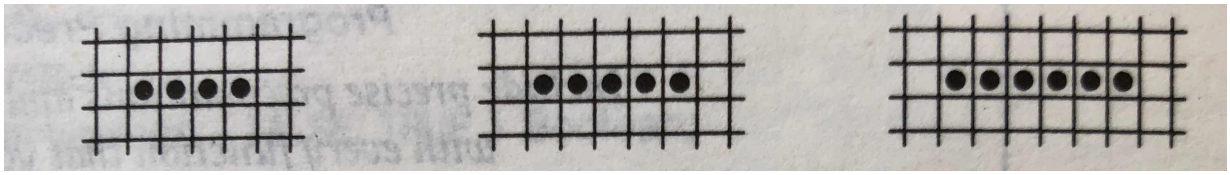
Figure 1(c) Alternation community

【Requirement】

Determine by hand calculation what will happen to each of the communities shown in Figure 2 over the course of three generations.

【Suggestion】

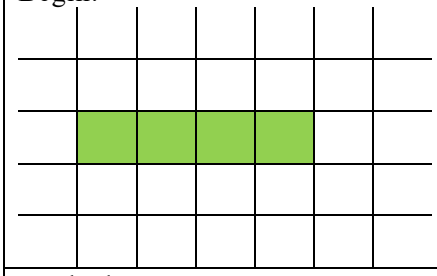
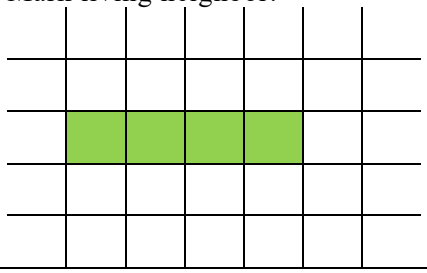
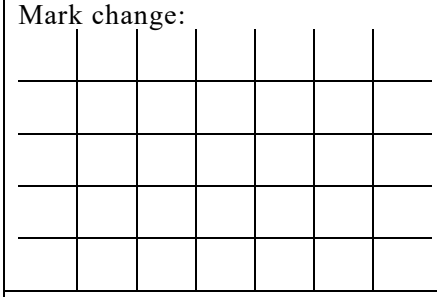
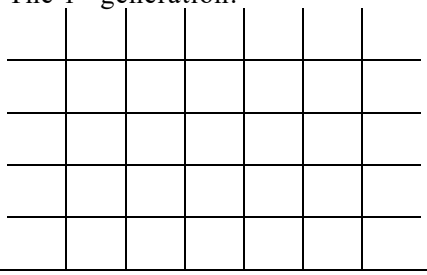
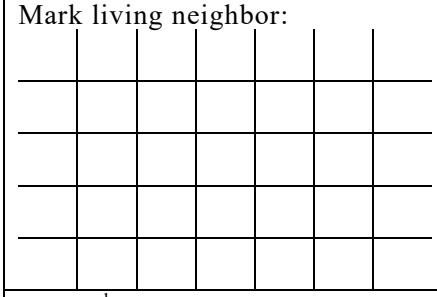
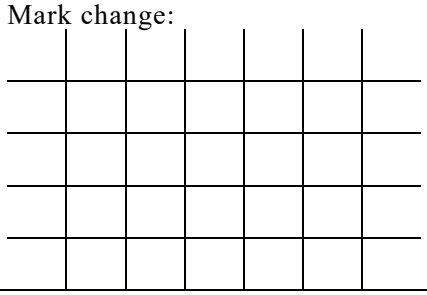
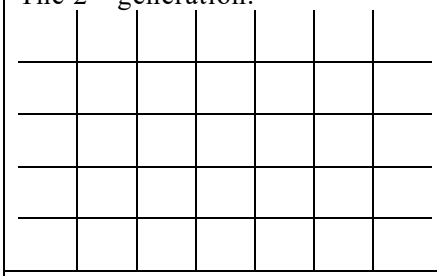
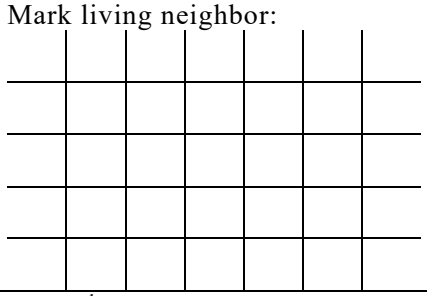
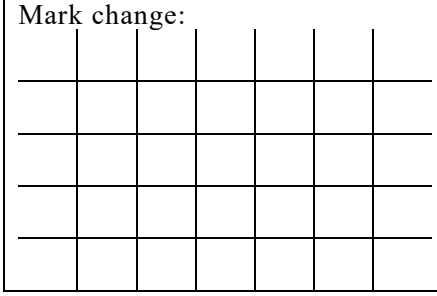
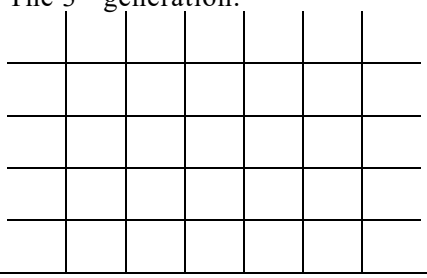
Set up the life configuration on a checkerboard. Use one color of checkers for living cells in the current generation and a second color to mark those that will be born or die in the next generation.



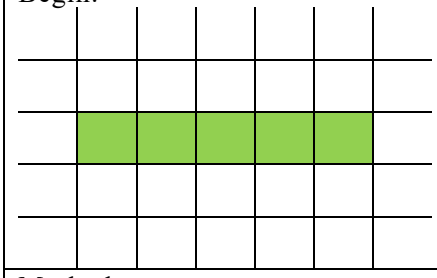
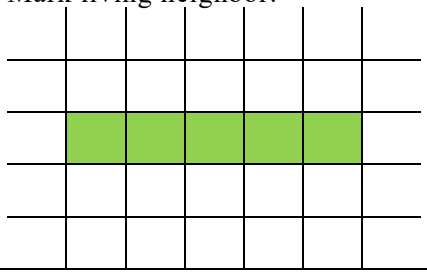
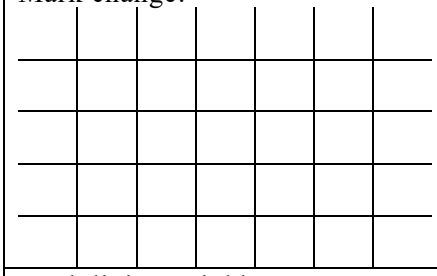
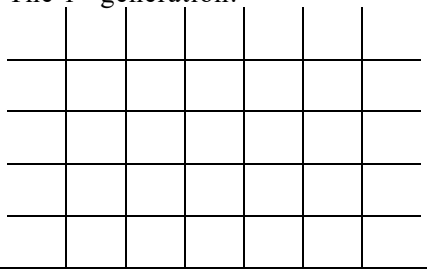
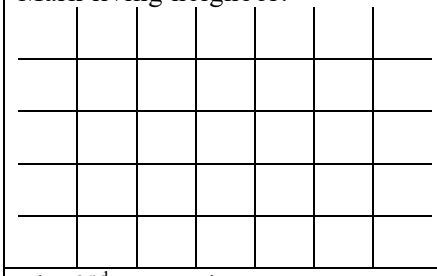
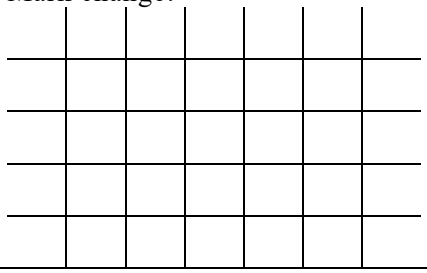
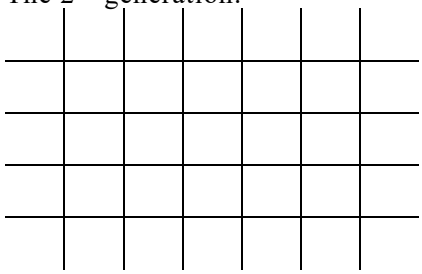
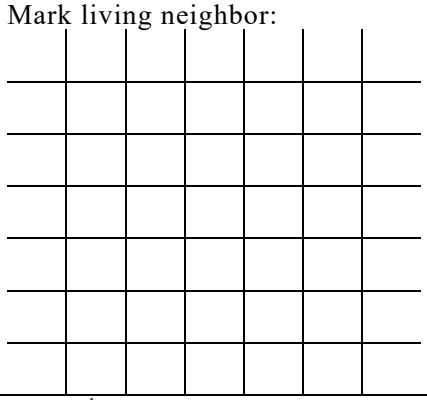
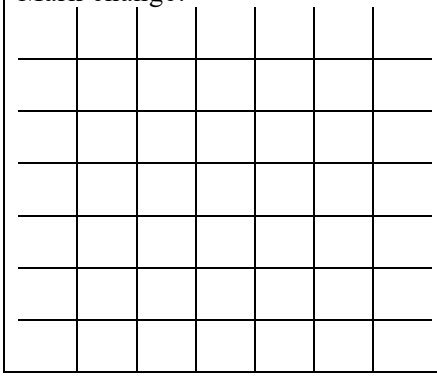
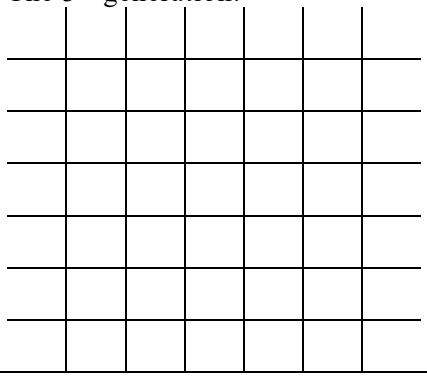
(a) **(b)** **(c)**

Figure 2 Simple Life configurations

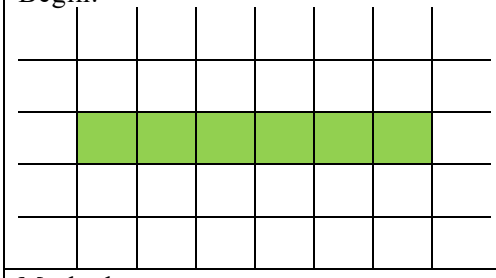
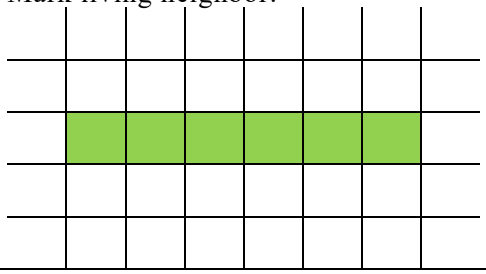
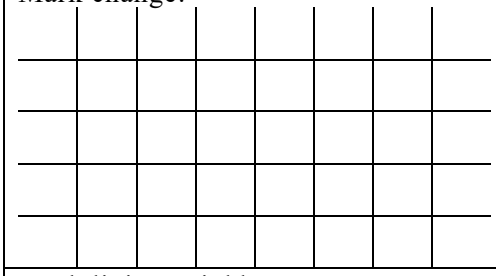
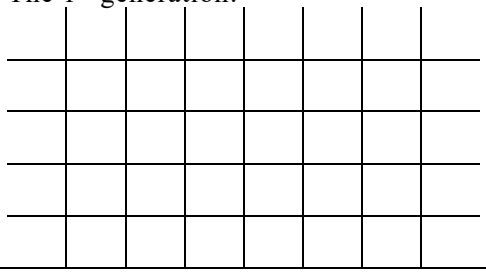
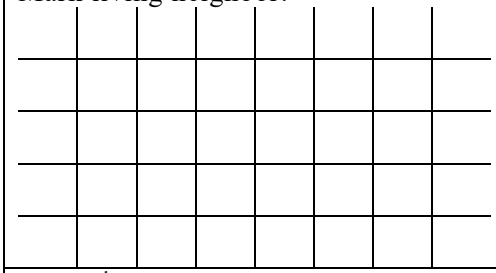
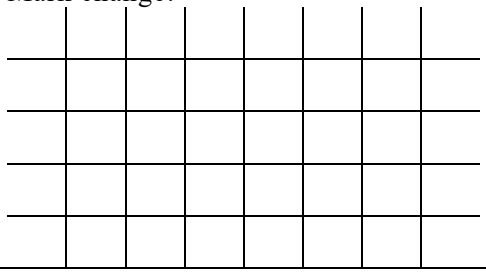
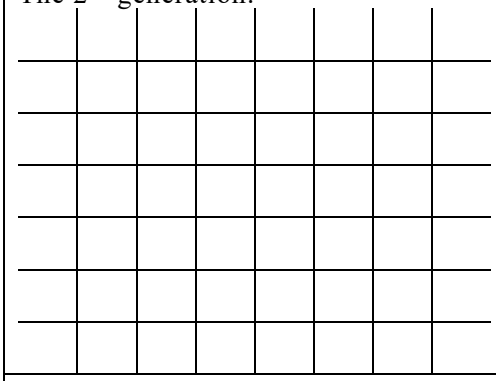
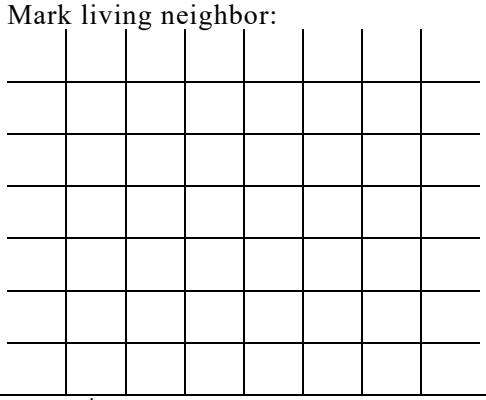
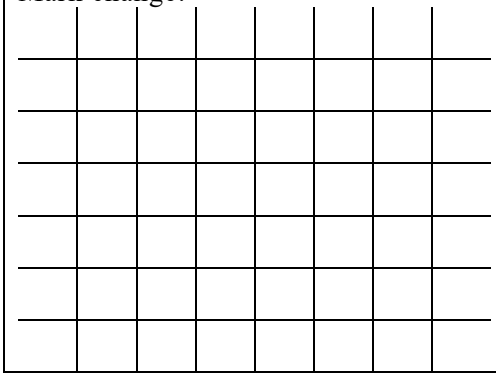
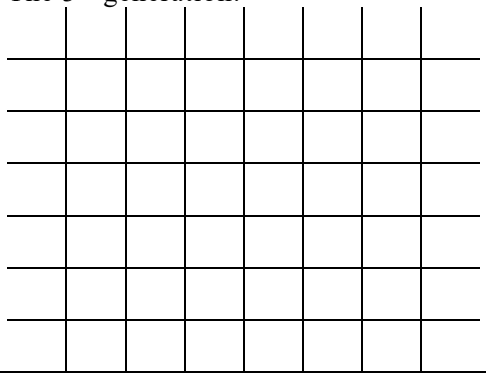
(a)

Begin: 	Mark living neighbor: 
Mark change: 	The 1st generation: 
Mark living neighbor: 	Mark change: 
The 2nd generation: 	Mark living neighbor: 
Mark change: 	The 3rd generation: 

(b)

Begin: 	Mark living neighbor: 
Mark change: 	The 1st generation: 
Mark living neighbor: 	Mark change: 
The 2nd generation: 	Mark living neighbor: 
Mark change: 	The 3rd generation: 

(c)

Begin: 	Mark living neighbor: 
Mark change: 	The 1st generation: 
Mark living neighbor: 	Mark change: 
The 2nd generation: 	Mark living neighbor: 
Mark change: 	The 3rd generation: 

Exercises 1.2 Choice Questions

1. The Run-Time Complexity of the following program segment is ( ).

```
for (i=0;i<n;i++)
    for (j=0;j<m;j++)    a[i][j]=0;
```

A. O(m) B. O(n) C. O(m*n) D. O(m/n)

2. The Run-Time Complexity of the following program segment is ( ).

```
i=1;        K=0;
While (i<=n-1)
        {k=k*10*i;     i++; }
```

A. O(n) B. O(1) C. O(n+1) D. $O(n^2)$

3. The Run-Time Complexity of the following program segment is ( ).

```
i=1;
do{ j=1;
    do{ printf("%d\n",i*j);      j++;          }while (j>n)
    i++;
}while (i>n);
```

A. O(n) B. O(1) C. $O(n^2)$ D. $O(n^3)$

4. The Run-Time Complexity of the following program segment is ( ).

```
I=0;        S=0;
While (s<n){    I++;          s=s+I;   };
```

A. O(n) B. $O(n^2)$ C. $O(n^{1/2})$ D. $O(n^3)$

5. The Run-Time Complexity of the Bubble Sort program is ( ).

A. O(n) B. $O(n^2)$ C. $O(n^3)$ D. $O(n^4)$

6. Assume the size of the instance is n, analysis the program below:

```
k=n;        m=0;            /* n>1    */
while (k>=(m+1)*(m-1))    m++;
```

The Run-Time Complexity of the following program segment is ( )

A. O(n) B. O(1) C. $O(n^{1/2})$ D. $O(n^2)$

7. Assume the size of the instance is n, analysis the program below:

```
a=10;   b=100;
while (b>0) {    a++;          b--;      }
```

The Run-Time Complexity of the following program segment is ( )

A. O(1) B. O(n) C. $O(n^2)$ D. $O(n^{1/2})$

Exercises 1.3 Draw logical form structures.

1. B1 = (D, R), D={1,5,8,12,20,26,34}, R={r},
   r={<1,8>,<8,34>,<34,20>,<20,12>,<12,26>,<26,5>}
   Please draw the logical form structure and write witch logical form it is.

2. B2 = (D, R), D = {a, b, c, d, e}, R={r},

r={(a, b), (a, c), (b, c), (c, d), (c, e), (d, e)}

Please draw the logical form structure and write witch logical form it is.

3. B3 = (D,R), D = {48,25,64,57,82,36,75}, R={r1,r2}

r1={<48,25>,<48,64>,<64,57>,<64,82>,<25,36>,<82,75>}

r2={<25,36>,<36,48>,<48,57>,<57,64>,<64,75>,<75,82>}

Please draw the logical form structure and write witch logical form they are.

Note: use two different line to represent the two relationship.

Exercises 1.4 Reorder the following efficiencies from smallest to largest:

a. $n\log(n)$ b. $n+n^2+n^3$ c. 24 d. $n^{0.5}$

Exercises 1.5 Determine the big-O notation for the following:

a. $5n^{5/2} + n^{2/5}$ b. $6\log(n) + 9n$ c. $3n^4 + n\log(n)$ d. $5n^2 + n^{3/2}$

Exercises 1.6 If the algorithm doIt has an efficiency factor of 5n, calculate the run-time efficiency of the following program segment:

```
for (i = 1; i <= n; i++)
    doIt (...)
```

Exercises 1.7 If the efficiency of the algorithm doIt can be expressed as $O(n) = n^2$, calculate the efficiency of the following program segment:

```
for (i = 1; i < n; i *= 2)
    doIt (...)
```

Exercises 1.8 Three students wrote algorithms for the same problem. They tested the three algorithms with two sets of data as shown below:

Case 1: n = 10

- •Run time for student 1: 1
- •Run time for student 2: 1/100
- •Run time for student 3: 1/1000

Case 2: n = 100

- •Run time for student 1: 10
- •Run time for student 2: 1
- •Run time for student 3: 1

What is the efficiency for each algorithm? Which is the best? Which is the worst?