1. (15%) Write an algorithm for converting infix expressions to postfix expressions.

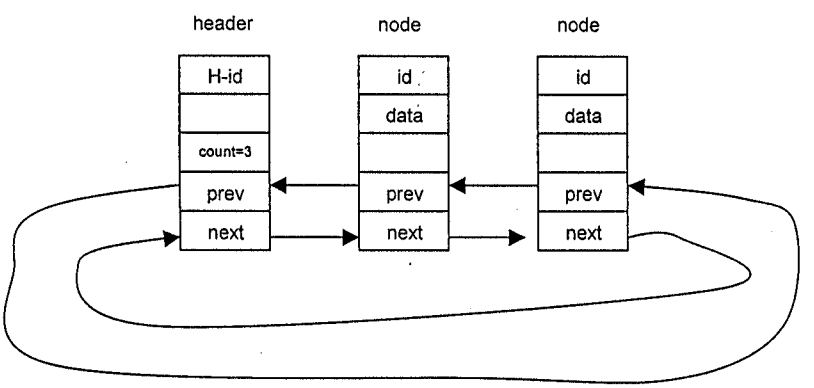
2. (15%) A heap is represented using an array, $X[1..13]=\{18, 9, 20, 6, 3, 10, 2, 8, 11, 21, 5, 12, 7\}$ Please utilize the Build-Heap algorithm to convert the heap into a max-heap. Show each step to receive full score.
Note: The division operation gets only the integer portion, e.g., 4 / 2 = 2, 5 / 2 = 2.

```
Build-Heap(A)
1   heap-size[A] ← length[A]
2   for i ← (length[A] / 2) downto 1
3       do Heapify(A, i)

Parent(i)
    return (i / 2)

Left(i)
    return (2 * i)

Right(i)
    return (2 * i + 1)
```

```
Heapify(A, i)      // A: array
                   // i: index of the array A
1   l ← Left(i)
2   r ← Right(i)
3   if l <= heap-size[A] and A[l] > A[i]
4      then largest ← l
5      else largest ← i
6   if r <= heap-size[A] and A[r] > A[largest]
7      then largest ← r
8   if largest != i
9      then exchange A[i] and A[largest]
10          Heapify(A, largest)
```

3. (5%) What are the advantages and disadvantages if we implement a heap using linked representation?

4. (15%) The following robust data structure consists of a header and nodes. A header contains id field, count field (number of nodes, including the header, in the list), and two pointers. Each node contains id field, data field, and two pointers. Using the robust linked list, we can detect errors when X (or less) failures occur (X-detectable) and can correct errors when Y (or less) failures occur (Y-correctable). Note: failures happen only at the count, prev, and next fields.
   (4-1) (5%) X=? Y=?
   (4-2) (10%) Explain your answer in detail.



5. (15%) Given an undirected graph G, how would you represent G in a computer program so that it is very efficient to find the cycle with the shortest path length among all the simple paths of G: (5 pts each)

(5-1) Describe your representation of G and give your reason.

(5-2) Show your algorithm (in any algorithmic language or programming language or a pseudo programming language) to find the cycle with the shortest path length.

(5-3) What is the time complexity of your algorithm given in (5-2)?

6. (15%) When realizing a hashing table of M entries, suppose we know *a priori* that a given key is equally likely to be any integer between k and $2k$, please answer the following and give your reason: (5 pts each)

(6-1) Under what conditions is the division method of hashing a good choice?

(6-2) Under what conditions is the middle-square method of hashing a good choice?

(6-3) Which collision resolution method of open addressing strategy is a good choice?

7. (20%) For each of the following questions, please pick all proper choices: (4 pts each)

(7-1) Which data structure(s) can be used to implement a comparison-based sorting algorithm?

(A) Stack (B) Priority Queue (C) Doubly linked list
(D) Network (E) Binary tree (F) Hash table

(7-2) Which is (are) true for bubble sort?

(A) unstable sorting algorithm (B) comparison-based sorting algorithm
(C) time complexity $O(n)$ (D) time complexity $\Omega(n)$
(E) space complexity $O(n^2)$ (F) space complexity $\Theta(n)$

(7-3) Which among the following sorting algorithms is (are) suitable for efficient implementation of external sorting method?

(A) Insertion sort (B) Quick sort (C) Heap sort
(D) Merge sort (E) Bucket sort (F) Radix sort

(7-4) Which among the following is (are) true?

(A) If $\forall n\ f(n) \geqq 0$ and $g(n) \geqq 0$, then either $f(n) = O(g(n))$ or $g(n) = O(f(n))$

(B) If $f(n) = O(g(n))$ and $g(n) = O(h(n))$, then $f(n) = O(h(n))$

(C) If $f_1(n) = O(g_1(n))$ and $f_2(n) = O(g_2(n))$, then $f_1(n) \cdot f_2(n) = O(g_1(n) \cdot g_2(n))$

(D) If $f(n) = O(g(n))$, then $\Omega(f(n)) = g(n)$

(E) Let $f(n) = \sum_{i=1}^{n} \log i$, then $f(n) = \Theta(n \cdot \log n)$.

(7-5) Which among the following is (are) true?

(A) AVL trees are binary search trees
(B) The height of an AVL tree of N nodes is $O(\log N)$
(C) A complete binary tree is not always height-balanced
(D) The height of a binary tree of N nodes is $\Omega(\log N)$
(E) B-Tree of order M is a height-balanced search tree