# Quiz 1

Name and student ID

- Write a simple C code for 2 dimensional array with dynamic memory allocation.

```
int i, j, **pi;
pi = (int **) malloc(sizeof(int*)*10);
if (pi == NULL) exit(1);

for (i = 0; i < 10; i++)
  pi[i] = (int *)malloc(sizeof(int)*20);
 // *(pi+i)
```

# Quiz 2

Name and student ID

- Write a C program that prints out the integer values of x, y, z in ascending order.

```
if (x < y) {
        if (y < z) printf ("%d %d %d\n", x, y, z);
        else if (x < z) printf ("%d %d %d\n", x, z, y);
        else printf ("%d %d %d\n", z, x, y);
} else {
        if (x < z) printf ("%d %d %d\n", y, x, z);
        else if (y < z) printf ("%d %d %d\n", y, z, x);
        else printf ("%d %d %d\n", z, y, x);
}
```

# Quiz 3

Name and student ID

- Show that the following statements are correct.
  - $5n^2 - 6n = O(n^2) \ \rightarrow \ n \geq 0,\ 5n^2 - 6n \leq 5n^2$
  - $\sum_{i=0}^{n} i^2 = O(n^3) \rightarrow \ n \geq 0, \frac{n(n+1)(2n+1)}{6} \leq n^3$
  - $n^3 + 10^6 n^2 = O(n^3) \ \rightarrow \ n \geq 10^6\ ,\ n^3 + 10^6 n^2 \leq 2n^3$
  - $\frac{6n^3}{\log n+1} = O(n^3) \ \rightarrow \ n \geq 1,\ \frac{6n^3}{\log n+1} \leq 6n^3$

# Quiz 4

Name and student ID

Let length[i] be the desired length of row i of a two dimensional array. Write a function similar to make2dArray() to create a two dimensional array such that row i has length[i] elements.

```
int i, j, **pi, noRow;
noRow = 5;
pi = (int **) malloc(sizeof(int*)*noRow);
if (pi == NULL) exit(1);

for (i = 0; i < noRow; i++)
  pi[i] = (int *)malloc(sizeof(int)*length[i]);
```

# Quiz 5

Name and student ID

Develop a structure to represent each of the following geometric objects: rectangle, triangle, and circle.

```
typedef struct {
   int x, y;
} coordinate;

typedef struct {
   int no_point;
   coordinate p[4];
} polygon;
```

# Quiz 6

Name and student ID

Implement Remove() and Attach() in slide 21 using polynomial representation in slide 22.

```
#define MAX_DEGREE 101
typedef struct {
        int degree;
        float coef[MAX_DEGREE];
} polynomial;
```

**A(x) = $3x^6 + 2x^5 + 41$**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| a | 6 | 3 | 2 | 0 | 0 | 0 | 0 | 41 |

degree

coef[]

```
 polynomial Attach(polynomial *p, float c, int e) {
   int diff;
   if (e <= p->degree) p->coef[p->degree+1-e] += c;
   else {
        diff = e - p->degree;
        for (i = e + 1; i >= diff + 1 ; i--)
           p->coef[i] = p->coef[i-diff];
        p->degree = e;
        p->coef[1] = c;
   }
}

Polynomial Remove(polynomial *p, int e) {
   int i = 1;
   p->coef[p->degree+1-e] = 0;
   if (e == p->degree) {
        while (p->coef[i] == 0) i++;
        for (j = 1; j <= p->degree + 1 – i; j++)
           p->coef[j] = p->coef[j+i];
        p->degree -= i;
}
```

# Quiz 7

Name and student ID

We have a 500 x 500 matrix. Find the smallest number of nonzero elements such that the memory space of 2-D array representation becomes less than that of the sparse matrix representation.

**<Solution>**

**2D array size = 500 x 500 x 4**
**Sparse matrix size = 3 x 4 x n**
**where n is the number of nonzero elements.**

**Therefore, the sparse matrix size becomes larger**
**When n >= 83,334.**

# Quiz 8

Name and student ID

Rewrite the following codes without rowTerms[] to save memory.

Hint: Reuse startingPos[].

```
for(i=1; i<= numTerms; i++){rowTerms[a[i].col]++;}
startingPos[0] = 1;
for(i = 1; i < numCols; i++){
        startingPos[i]=startingPos[i-1]+rowTerms[i-1];}
```

<Solution>

```
for(i=1; i<= numTerms; i++){startingPos[a[i].col]++;}
temp0 = startingPos[0];
startingPos[0] = 1;
for(i = 1; i < numCols; i++){
       temp1 = startingPos[i];
       startingPos[i]=startingPos[i-1]+temp0;
       temp0 = temp1;
}
```

# Quiz 9

Name and student ID

Write the string remove function to remove j characters beginning from i in string s.

**<Solution>**

```
void stringremove (char *s, int i, int j) {
    len = strlen(s);
    if (len < i+j) {
        fprintf (stderr, "position is out of bounds\n");
        exit(1);
    }
    for (x = i; x <len - j; x++)
        s[x] = s[x+j];
    s[x] = '\0';
}
```

# Quiz 10

Name and student ID

Write a function strnchar() that takes a string and a character as input, and returns the number of occurrences of the character in the string.

```
int strnchar(char *s, char p) {
    int i, n, p_count;
    n = strlen(s);
    p_count = 0;
    for (i = 0; i < n; i++) {
         if (s[i] == p) p_count ++;
    }
    return p_count;
}
```

# Quiz 11

Name and student ID

Compute the failure function for each of the following patterns.

**(a) aaaab: -1 0 1 2 -1**

**(b) ababaa: -1 -1 0 1 2 0**

**(c) abaabaab: -1 -1 0 0 1 2 3 1**

# Quiz 12

Name and student ID

Rewrite pmatch() in slide 12 using for loop instead of while loop.

```
int pmatch (char* string, char* pat)
{
  int  i=0; int j=0;
  int lens=strlen(string); int lenp=strlen(pat);
  for (i=0, j=0; i < lens && j < lenp;){
    if (string[i] == pat[j]) { i++; j++; }
    else if (j == 0) i++;
    else j = failure[j-1]+1;
  }
  return (j == lenp ? i-lenp : -1);
}
```

# Quiz 16

Name and student ID

Suppose a slightly modified maze problem such that we can move to only four directions (North, East, South, and West). Then, how do we need to modify path() ?

```
while (dir < 4 && !found) {
    /* move in direction dir*/
    nextRow = row + move[dir].vert;
    nextCol = col + move[dir].horiz;
    if (nextRow==EXIT_ROW && nextCol==EXIT_COL)
        found = TRUE;
    else if ( !maze[nextRow][nextCol]
                 && !mark[nextRow][nextCol]) {
        mark[nextRow][nextCol]) = 1;
        position.row = row; position.col = col;
        position.dir = ++dir;
        push(position);
        row = nextRow; col = nextCol; dir = 0;
    }
    else ++dir;
 } /* while (dir < 4 & !found)
```

# Quiz 17

Name and student ID

Evaluate the following postfix expressions.
(assume that all the operands are single digit)

(a) 7 3 2 + * 3 – = 32

(b) 3 4 5 * 4 / + = 8

(c) 9 3 * 4 5 * + 3 - = 44

# Quiz 18

Name and student ID

Write the postfix form of the following expressions:

(a) a * b * c => a b * c *

(b) -a + b – c + d => -a b + c – d +

(c) a * -b + c => a –b * c +

(d) (a + b) * d + e / (f + a * d) + c => a b + d * e f a d * + / c +

# Quiz 19

Name and student ID

Modify delete() function in the previous slide so that it takes only two arguments, first and x without trail.

```
void delete(listPointer *first, listPointer x) {
{
    listPointer trail;
    if (*first == x)
        *first = (*first)->link; // deleting the first node
    else {
        trail = first;
        for ( ;trail->link != x; trail = trail->link);
        trail->link = x->link;
    }
    free(x);
}
```

# Quiz 20

Name and student ID

In slide 17, modify addq() with the assumption that we have only front not rear.

```
void addq (int i, element item)
{  /* insert an item at the rear of queue i */
   queuePointer temp, rear;
   MALLOC(temp, sizeof(*temp));
   temp->data = item;
   temp->link = NULL;
   if (front[i]) {
        for (rear = front[i]; rear->link != NULL;
                                  rear = rear->link);
        rear->link = temp;
   }
   else front[i] = temp;
}
```

# Quiz 21

Name and student ID

In the previous slide, modify erase() using for loop instead of while().

```
void erase(polyPointer* ptr) {
   polyPointer temp;
   for ( ; *ptr != NULL; ) {
      temp = *ptr;
      *ptr = (*ptr)->link;
      free(temp);
   }
}
```

# Quiz 22

Name and student ID

In the previous slide, explain how to modify cpadd() if there is no special node indicating the first node. You do not need to submit the actual code.

**>> If there is no special node (with exponent '-1') in a circular list, we have to check if we reach the first node whenever we move to the next node.**

# Quiz 23

Name and student ID

Implement a function, invertedCopyList(), for a given list, to copy the list, invert it, and return.

```
listPointer invertedCopyList(listPointer ptr){



}
```

>> You can easily implement by slightly modifying invert().

# Quiz 24

Name and student ID

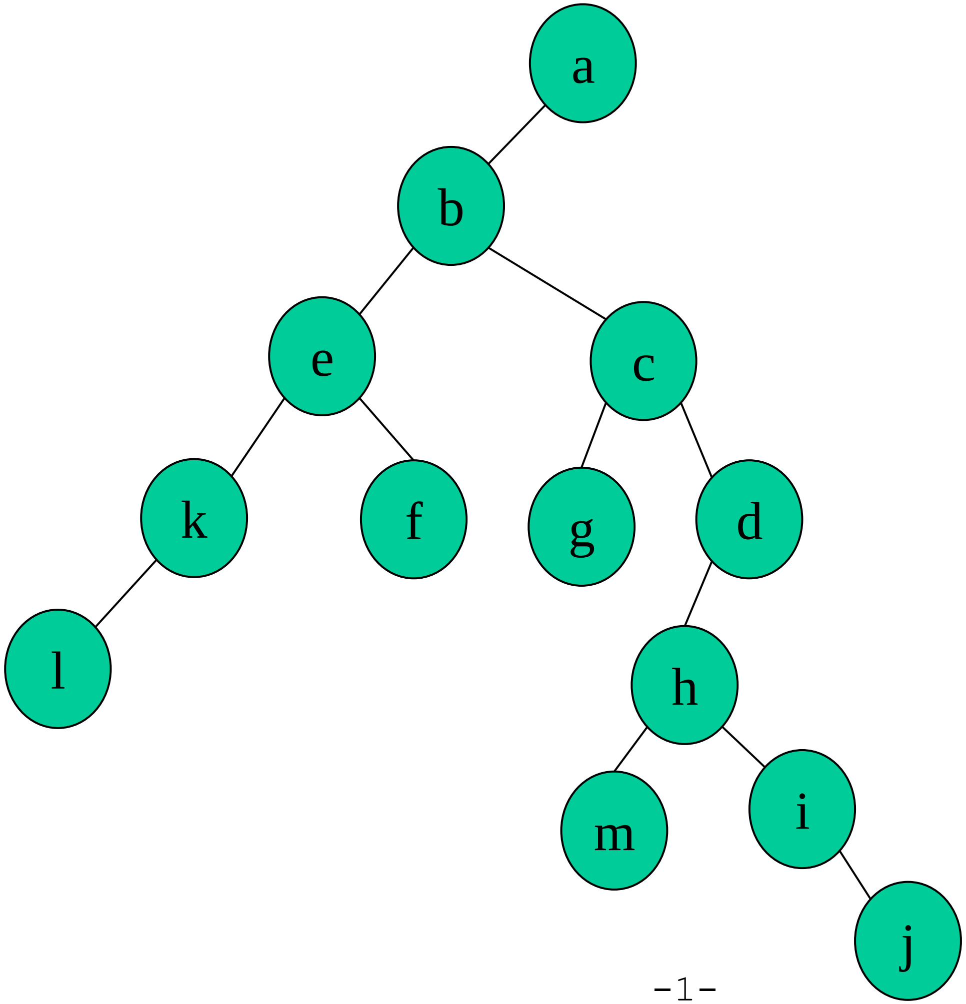
```
count = 0;
for (i = 0; i < n; i++)
    if (out[i]) {
       count ++;
       printf("\nNew class: %5d", i);
       out[i] = FALSE; x = seq[i]; top = NULL;
       for (;;) {
           while (x) {
             j = x->data;
              if (out[j]) {
               printf(%5d", j);   out[j] = FALSE;
               y=x->link; x->link=top; top=x; x=y;
             } else x = x->link;
           }
           if (!top) break;
           x = seq[top->data];
           top = top->link;
        } /* for (;;) */
} /* for (i)  */
printf("%d\n", count);
} /* main() */
```

>> Since out[i] is initialized as TRUE for 0<= i <n, count simply becomes ‘n’.

# Quiz 25

Name and student ID

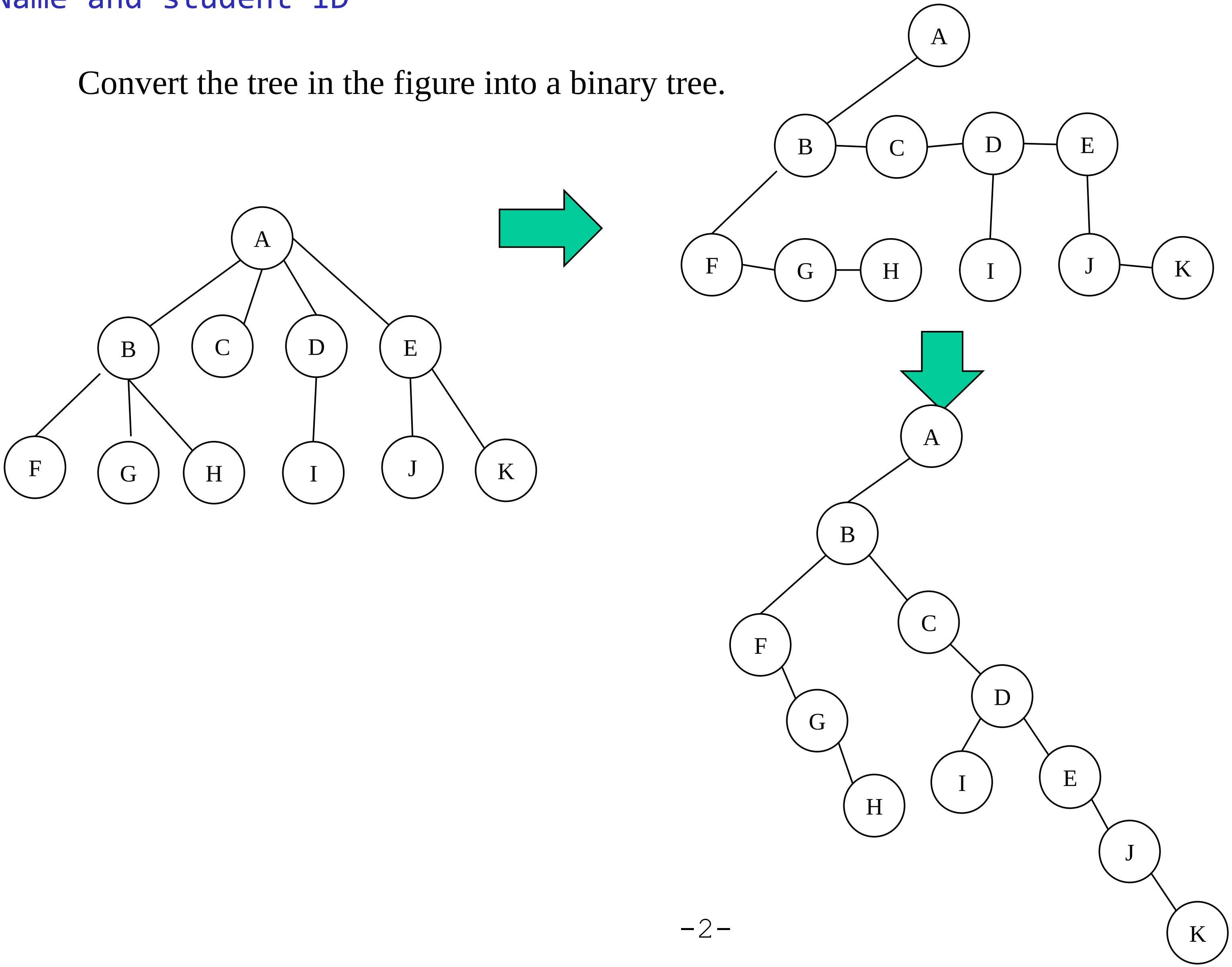
Draw the tree given as the list (a(b(e(k(l),f),c(g,d(h(m,i(j))))))).



# Quiz 26

Name and student ID

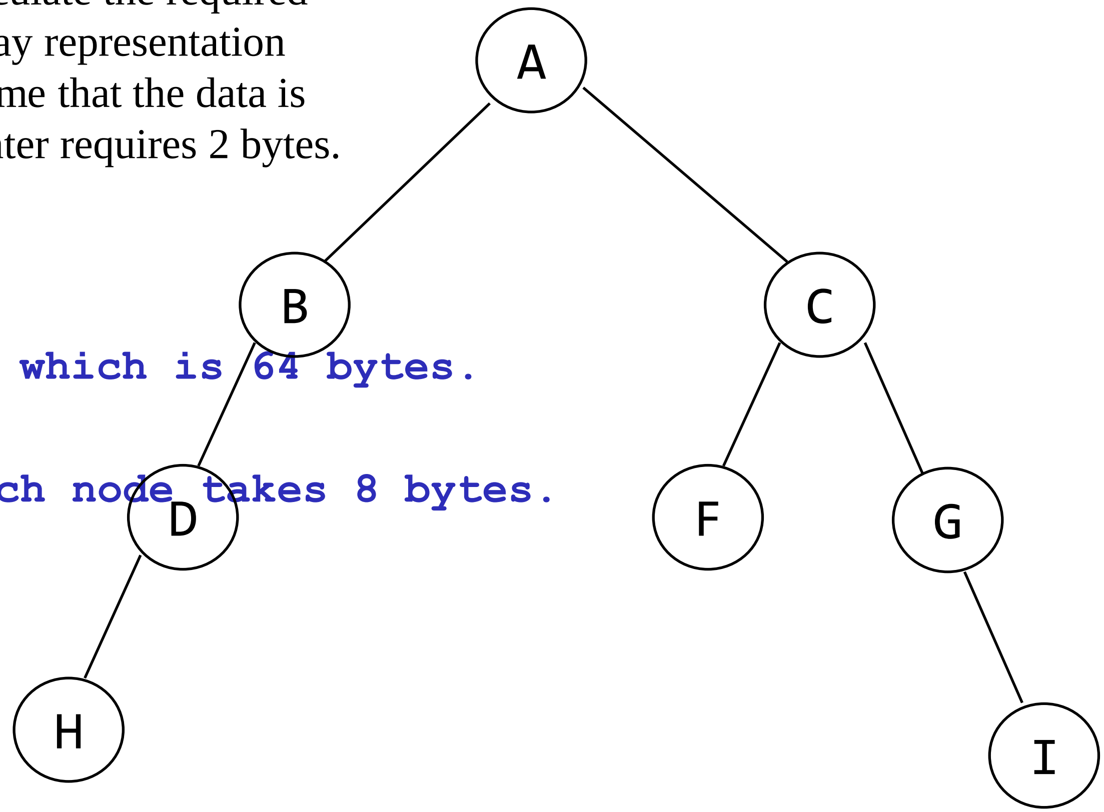
Convert the tree in the figure into a binary tree.



# Quiz 27

Name and student ID

Consider the tree in the figure. Calculate the required memory space when we use (a) array representation and (b) linked representation. Assume that the data is integer (4 bytes), and the node pointer requires 2 bytes.




**(a) Array representation**

**We need 16 integer space, which is 64 bytes.**

**(b) Linked representation**

**There are 8 nodes, and each node takes 8 bytes.**

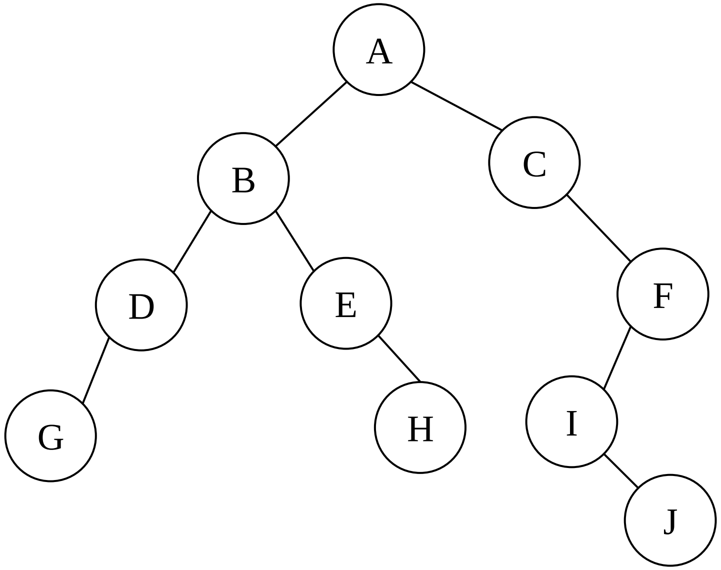
**So, we need 64 bytes.**

# Quiz 28

Name and student ID

Write out the inorder, preorder, postorder, and level-order traversals for the given binary tree.

Inorder: G D B E H A C I J F
Preorder: A B D G E H C F I J
Postorder: G D H E B J I F C A
Level-order: A B C D E F G H I J



# Quiz 29

Name and student ID

Write a function treeAdd() which returns the sum of data in a tree.

```
typedef struct node* treePointer;
typedef struct node {
    int data;
    treePointer leftChild, rightChild;
};
```

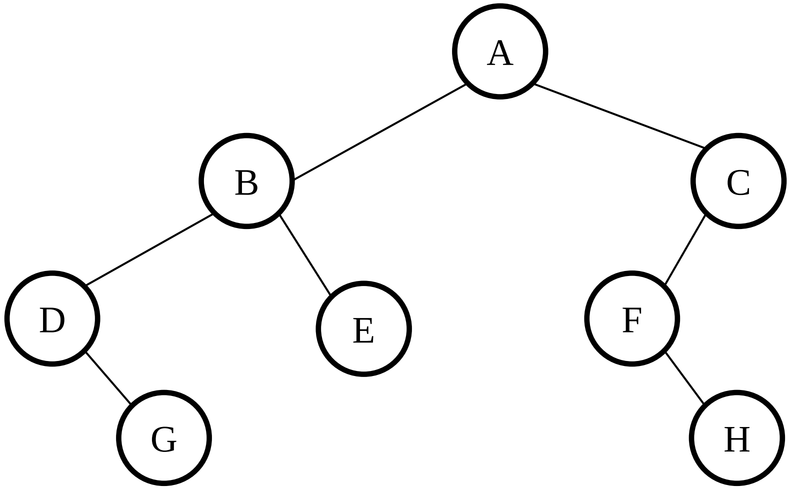
```
int treeAdd(treePointer tree) {
  int sum = 0;

  if (ptr) {
    sum = treeAdd(ptr->leftChild);
    sum += ptr->data;
    sum += treeAdd(ptr->rightChild);
  }
  return sum;
}
```

# Quiz 30

Name and student ID

Rebuild a binary tree from the following traversal results:

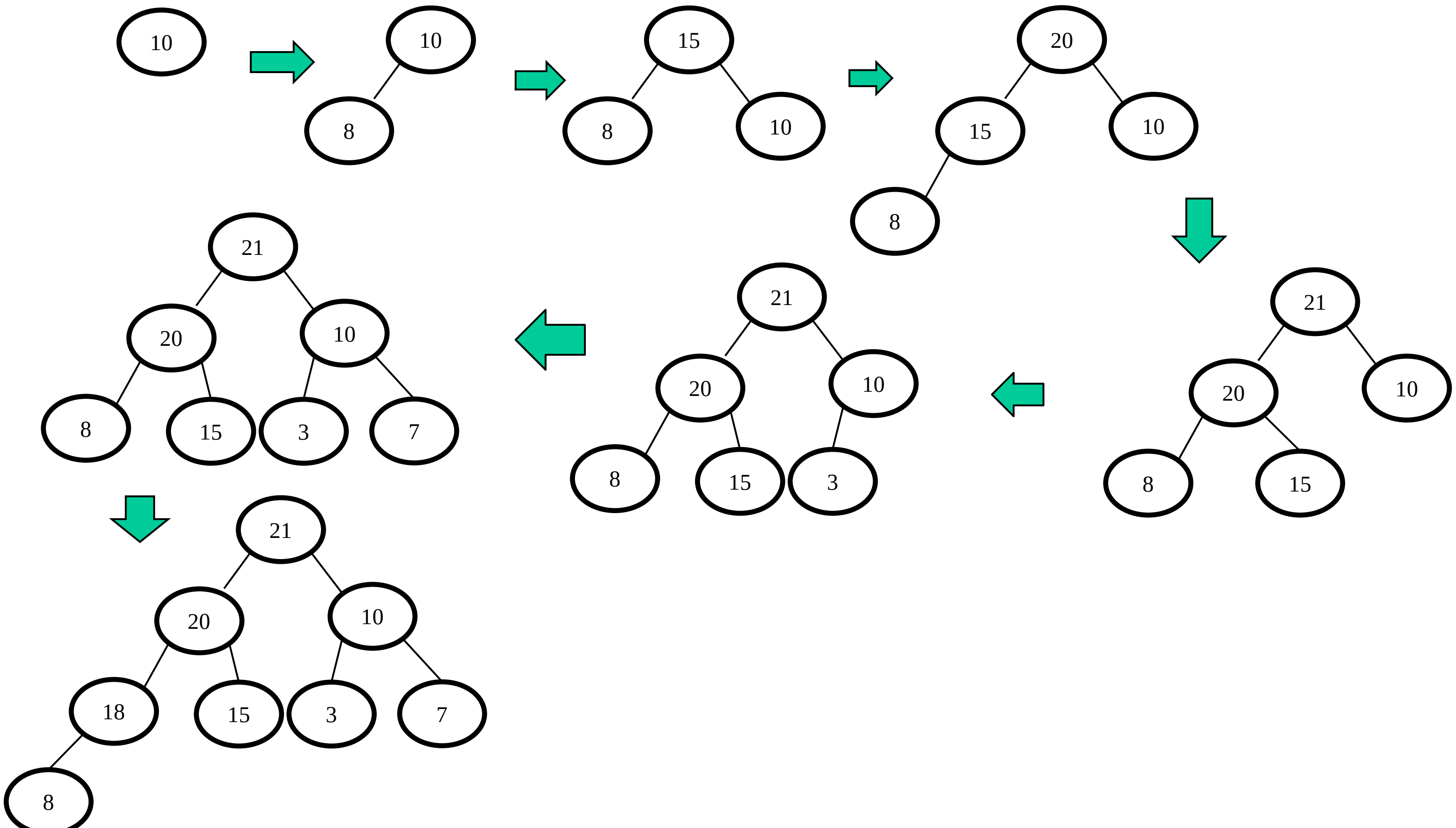
Inorder traversal: D G B E A F H C
Postorder traversal: G D E B H F C A



# Quiz 31

Name and student ID

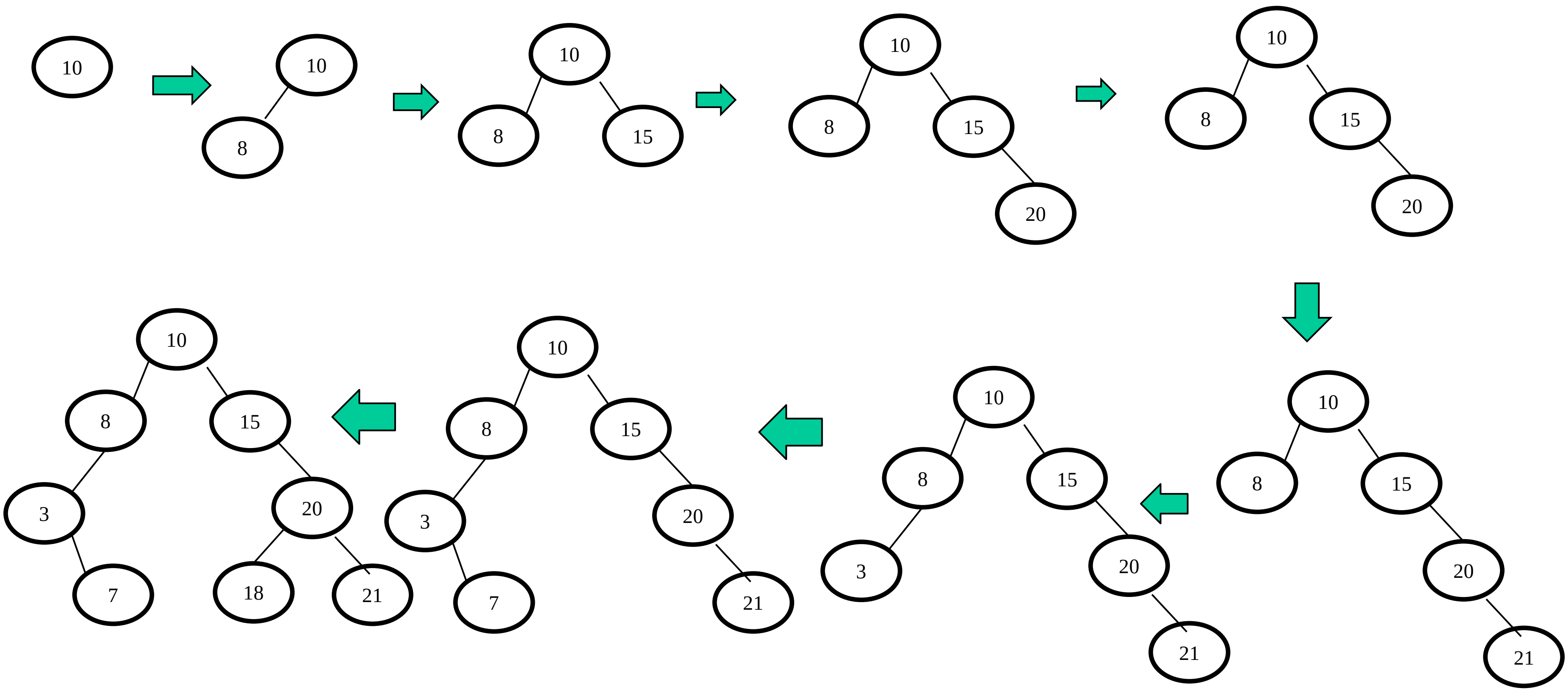
Draw the max heap after inserting 10, 8, 15, 20, 21, 3, 7, 18 from an empty heap.



# Quiz 32

Name and student ID

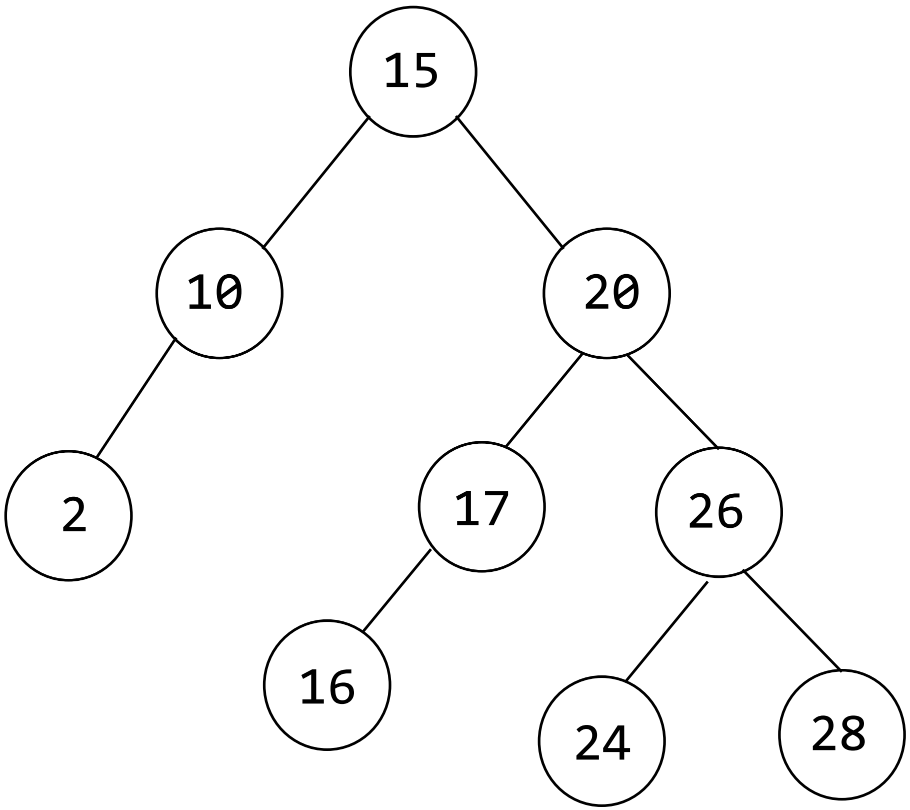
Draw the BST after inserting 10, 8, 15, 20, 21, 3, 7, 18 from an empty BST.

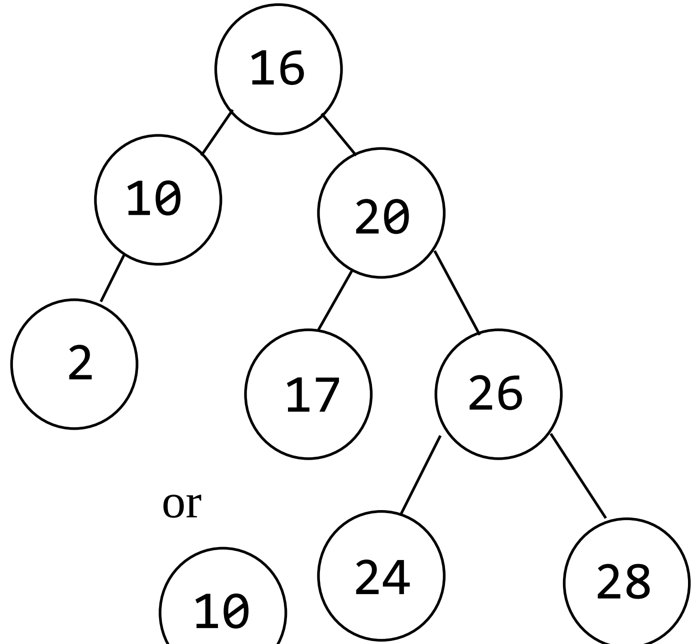


# Quiz 33

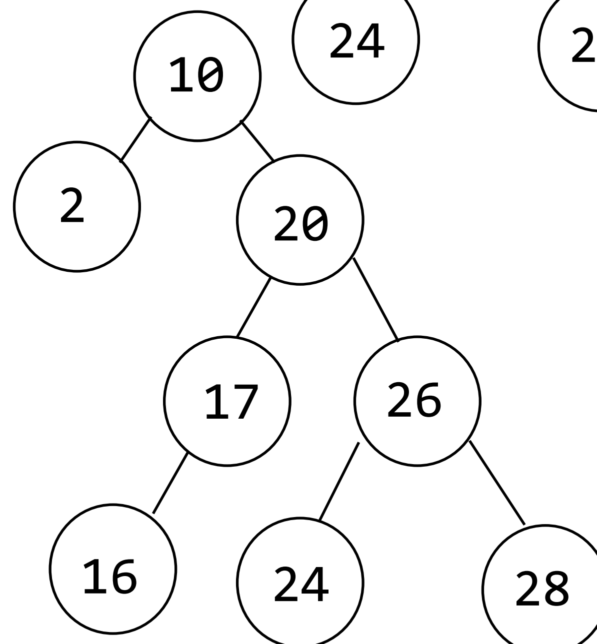
Name and student ID

Draw the resulting tree after deleting 15.




or

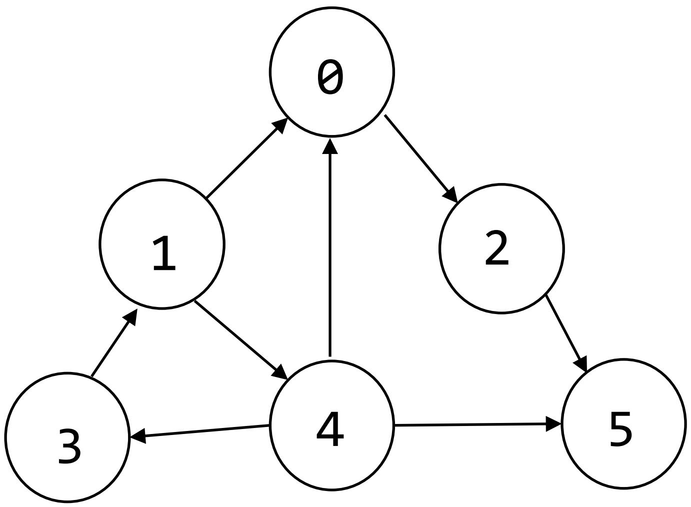


# Quiz 34

Name and student ID

Draw the strongly connected components of the following graph.

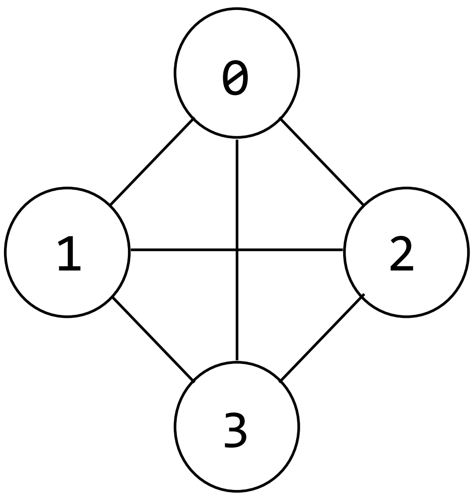
**Since there is a path between all the pairs in the graph, the graph is strongly connected.**



# Quiz 35

Name and student ID

Calculate the required memory to represent the following graph using adjacency matrix, adjacency list, and sequential representation, respectively.

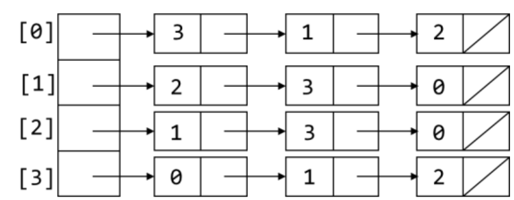



Adjacency matrix: 4 x 4 x 4 byte = 64 bytes

$$\begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

Adjacency list: 8 + 72 = 80 bytes

- 4 Head pointers (4 x 2 bytes)
- 12 nodes (12 x (4 + 2) bytes)




Sequential representation: (n + 2xe + 1) x 4 byte

- n = 4, e = 6
- (4 + 2x6 + 1) x 4 byte = 68 bytes

# Quiz 36

Name and student ID

Show the adjacency multilist for the following graph (slide 32).