Name: David Huynh ID#

COMPUTING II

MIDTERM 1

You have 50 minutes to complete the midterm. The total number of points is 60. You are not allowed to use any books, notes, calculator, or electronic devices. Write your answer carefully and clearly. Incorrect answers will receive little to no points. When you are asked to write a program the program should be clear and include indentations and comments to make it easier to read. ***Be sure to state any assumptions on which you have based your answers.***

Problem Number(s)	Possible Points	Earned Points
1	7	7
2	7	7
3	10	10
4	10	8
5	6	2
6	15	5
7	15	1
8	2 ***extra***	0
	TOTAL POINTS 70	40

1. **Stack** (7 points)
Draw a sequence of diagrams, one for each problem segment, that represent the current state of the stack after each labeled set of operations. If an operation or instruction produces output then indicate what that output is. **hStack** is the handle of a stack opaque object that can hold characters.

```
hStack = stack_init_default();

a. stack_push(hStack, '1');
b. stack_push(hStack, '2');
c. printf("%c ", stack_top(hStack)); stack_pop(hStack);
d. printf("%c ", stack_top(hStack)); stack_push(hStack, '3');
e. stack_push(hStack, '4');  stack_push(hStack, '5');
f. printf("%c ", stack_top(hStack)); stack_push(hStack, '6');
g. stack_pop(hStack); stack_push(hStack, '5');

stack_destroy(&hStack);
```

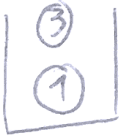
a/ 

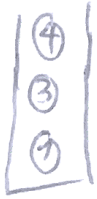
b/ 

c/ - ~~output~~ print out put : 2

- 

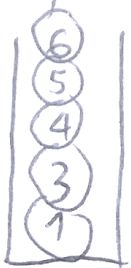
d/ - print output: 1

- 

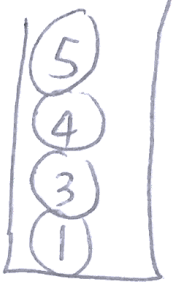
e/ - ~~print output:~~ 3 

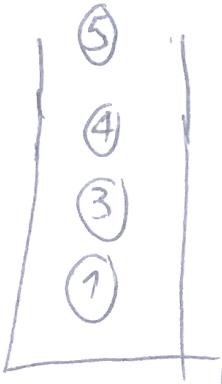
tLen

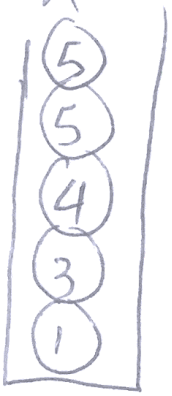
f/ - print out put : 5 



2
tLen

g/ - 



tLen


2. **Queues** (7 points)
Draw a sequence of diagrams, one for each problem segment, that represent the current state of the queue after each labeled set of operations. If an operation or instruction produces output then indicate what that output is. We will use the function enqueue to add to the queue and serve to remove from the queue. **hQueue** is the handle of a queue opaque object that can hold characters.

```
hQueue = queue_init_default();
```

```
a. queue_enqueue(hQueue, 'a');
b. queue_enqueue(hQueue, 'b');
c. printf("%c, " queue_front(hQueue)); queue_enqueue(hQueue, 'c');
d. printf("%c ", queue_front(hQueue)); queue_enqueue(hQueue, 'd');
e. queue_serve(hQueue); queue_serve(hQueue);
f. printf("%c ", queue_front(hQueue)); queue_enqueue(hQueue, 'e');
g. queue_serve(hQueue); queue_serve(hQueue);
```

```
hQueue->destroy(&hQueue);
```

a/ [a]

b/ [b] → [a]

c/ print: ~~b~~ a then [c] → [b] → [a]

d/ print: ~~a~~ a then [d] → [c] → [b] → [a]

e/ - [d] → [c] → [b]

- [d] → [c]

f/ print c; then [e] → [d] → [c] ✓

g/ [e] → [d]

- [e]

3. **Expression Evaluation** (10 points).
Evaluate the following expressions assuming 32 bit integers and 32 bit pointers. Variables are declared as listed but after some unknown number of operations the current state of the memory is given by the supplied memory diagram.

```
struct node
{
      int data;
      struct node* other;
};
typedef struct node Node;

Node v;
Node* p;
```

Variable Name /	Address	Memory Value
v	8000	3
	8004	9000
	8008	9004
p	8012	9028
	8016	9032
	8020	9020
	...	...
	9000	74
	9004	9016
	9008	5
	9012	100
	9016	87
	9020	9008
	9024	101
	9028	1
	9032	8000
	9036	9016

a. `v.other;` 9000

b. `v.other->data;` 74

c. `(p->other->data) << 3;` 24

d. `(p->other->other->data) + 3;` 77

e. `p->other->other->other->other` 9008

linked list

The next declaration applies to Question 4, 5, and 6:

```
typedef struct node Node;
      struct node
      {
            int data;
            Node* next;
      };
```

4. (10 points) Write a function called **destroy** that takes a **Node pointer** to the head of a list and will free up the memory associated with each node in the entire list.

8

```
void destroy (Node** pHead ) {
    Node* temp;
    temp = *pHead;
    while (temp != NULL){
        *pHead = temp->next;
        free (temp);
        temp = *pHead;
    }
}
```

5. (6 points) Write a recursive function called sum that given a Node pointer to the head of a list will return the sum of all data in the linked list.

   (3 points for iterative solution)

```
int* Sum(Node* head, ) {
int sum=0.
Node *temp ; *next ;
temp = head;
while(temp != NULL {
return item temp
next = temp -> next ;
return: sum temp Sum(temp
}
```

+2

6. (15 points) Write a function called **`copy_list`** that, given a ***Node pointer*** to the head of a list will return a ***Node pointer*** containing the address of the head node of a new list that is an exact copy of the original list. Your copy should be independent of the first list and not share any nodes. You may write an iterative or recursive version of your function.

```
Node* copy_list( Node* head ){
Node *temp, *next; *copy;
temp = head;
while (temp != NULL ){
~~Next~~ temp
```

7. **Vector** (15 points) In class we created an ***opaque object*** for a type called **VECTOR** that had an internal structure called **Vector** consisting of an integer size, an integer capacity, and an integer pointer data that held the address of the first element of a dynamic array of integers. Write a function called **vector_init_default( )** that initializes the vector to have a size of zero, capacity of seven and an appropriate value in the data pointer. ***Your function should return the address of an opaque object upon success and NULL otherwise***.

```
typedef struct node Node;
struct Node {
int size;
int capacity;
int* data;
}
```

8. (2 ***extra*** points)
   a. What is an opaque handle?

   b. What is an Abstract Data Type?

like Stack:
- container of objects that are inserted and removed according to the LIFO principle.

+ 0

Name: David Huynh ID#

COMPUTING II

MIDTERM 2

You have 50 minutes to complete the midterm. The total number of points is 70. You are not allowed to use any books, notes, calculator, or electronic devices. Write your answer carefully and clearly. Incorrect answers will receive little to no points. When you are asked to write a program the program should be clear and include indentations and comments to make it easier to read. ***Be sure to state any assumptions on which you have based your answers.***

Problem Number(s)	Possible Points	Earned Points
1	8	0
2	20	6
3	8	4
4	15	15
5	5	5
6	10	6
7	4	2
8	2 ***extra***	0
	TOTAL POINTS 70	38

Complexity

1. (8 points) Queue has the following operations: create Queue, insert Item, remove Item, destroy Queue.

   We considered two underlying implementations of the queue by using array and a single linked list. Complete the following table using Big O notation to denote the complexity of each operation using the given implementation strategy on the left. Please assume the next:

   a. the array will not need a resize operation during an insert or remove for this exercise
   b. we use pointers to head and tail of the Queue

	Create	Insert	Remove	Destroy
Array				
Singled Linked List				

Sorting Algorithms

2. (20 points) Define the following sorting algorithms using text and/or pseudo code.

(A.) Selection Sort

```
for (i=0 ; i< length ; i++ ){
    for(j=0 , j < i ; j++ ){
        if (arr[j] < arr[i])
            swap(arr[i], arr[j]) ;
} }
```

? -5

(B.) Bubble Sort

~~1st for int i~~

~~for(int i=0 , i < length-1 ; i++)~~

~~for(j=i ; i ;<~~

~~- Select the pair~~

-2

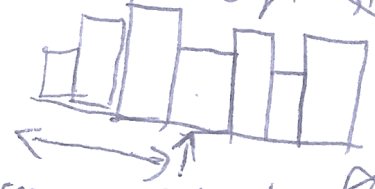
- start with the 1st elemt. pair with the next elemnt to compare. if ~~is~~ less than the next one ⇒ swap
  ~~1st~~ greater

~~Compare~~ Then ~~do the same~~ start with the next elemnt and pair with the next element to compare.
if its greater than the next one ⇒ swap

(C.) Insertion Sort

start select the ~~fist~~ element from left to ~~th~~ right compare its value with other elements on the left side of its. If it smaller -> swap

ex:



compare select

(1)

~~Recursive~~: repeat the step until select the last one on the right.

(D.) Shell Sort

~~fist~~ - select the gap by using the length ~~de~~

then compare the value ~~each in the each~~ between 2 gap the gap

For ~~ex~~ example gap = 2 :

if ~~it~~ the value smaller we swap it.

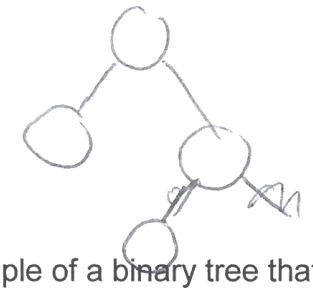
~~Recursiv~~ recursive until gap = 1

(2)

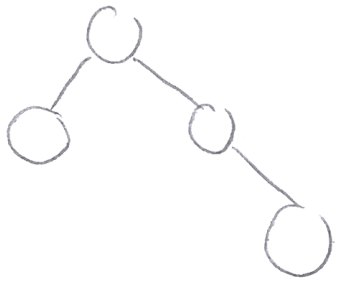
Tree

3. (8 points) Full vs Complete binary trees

   a. Give an example of a binary tree that is full but not complete.

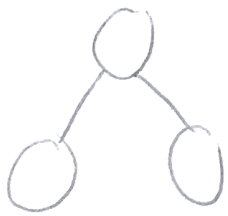


   b. Give an example of a binary tree that is complete but not full.

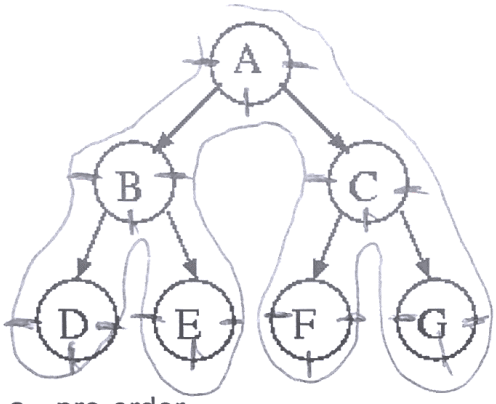


   c. Give an example of a binary tree that is neither full nor complete.

   d. Give an example of a binary tree that is both full and complete.



4. (15 points) List the sequence of nodes visited by pre-order, in-order, and post-order traversals of the following tree:



a. pre-order

A B D E C F G ✓

b. in-order

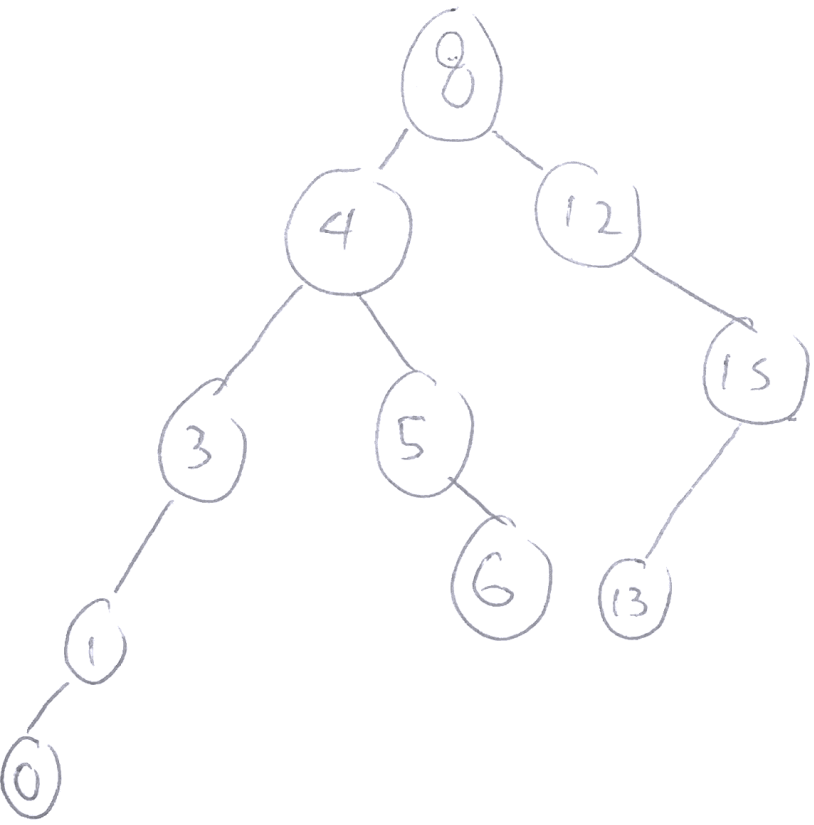
D B E A F C G ✓

c. post-order

D E B F G C A ✓

5. (5 points) Assuming the following values arrive in the order they appear and are inserted into a ***Binary Search Tree***, show the resulting tree.

8, 12, 4, 15, 3, 5, 13, 1, 0, 6



6. (10 points) Write down a recursive function to destroy a tree. The function takes a node pointer. The recursive function declaration is:

```
void binary_tree_destroy(BinaryNode pNode);
```

```
Void binary_tree_destroy(BinaryNode pNode){
    if(pNode != Null){
        binary_tree_destroy(pNode->left);
        binary_tree_destroy(pNode->right);
    }
    free(pNode);
    pNode = Null;
}
```

-4

AVL tree

7. (4 points) Build the AVL tree using the following data:

   2, 3, 5, 7, 9, 10, 8

8. (2 extra points) What is the time complexity of following code:

```
int a = 0;
for (i = 0; i < N; i++) {
    for (j = N; j > i; j--) {
        a = a + i + j;
    }
}
```

N = 5

i = 0; i < 5; i++

j = 5; j > i; j--

a = 0 + 0 + 5 + 1

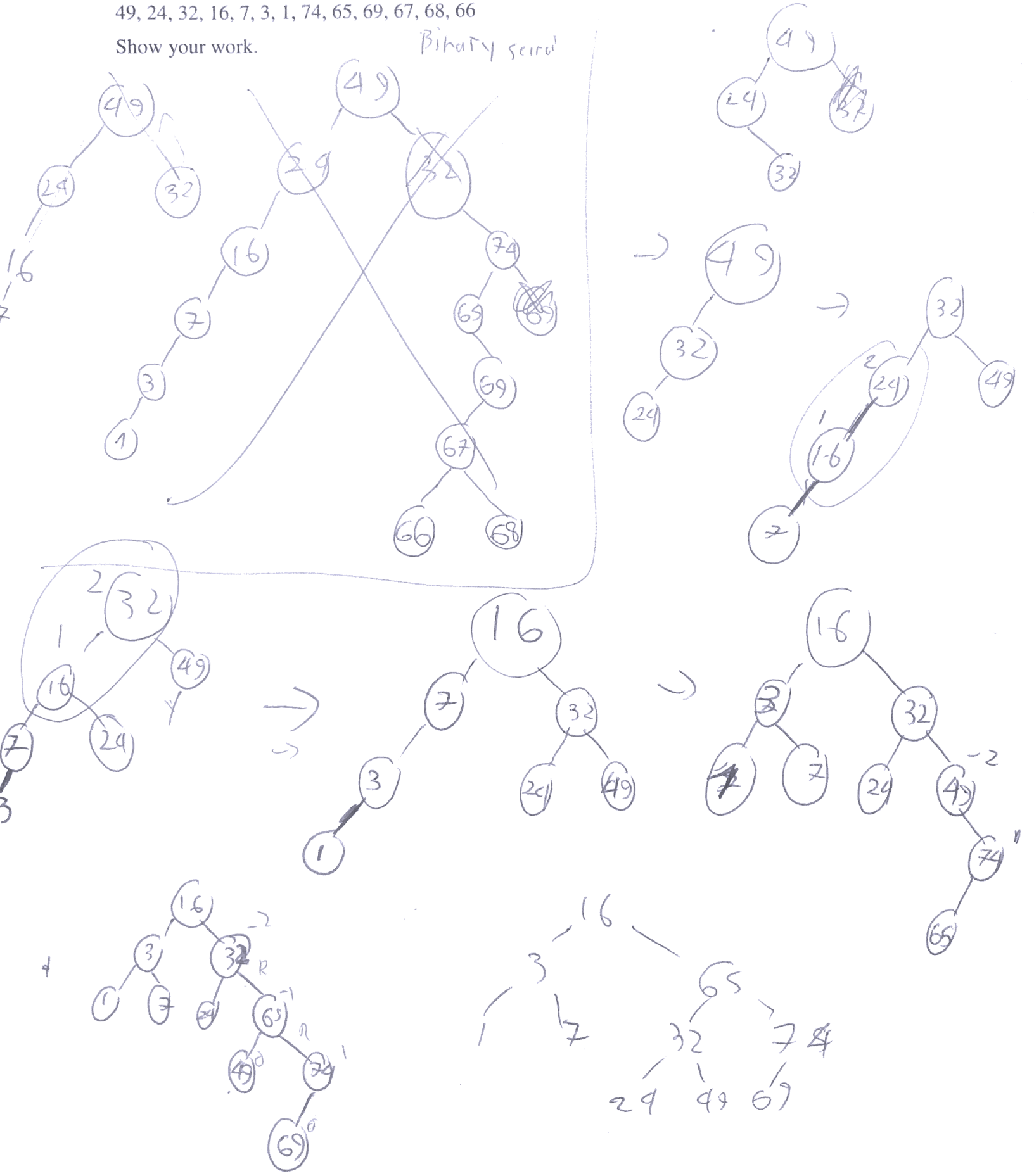
a =

a is sum of the loop for. ? x

Build the ***AVL tree*** by inserting the following numbers one by one:

49, 24, 32, 16, 7, 3, 1, 74, 65, 69, 67, 68, 66

Show your work.



Name: ID#

COMPUTING II

Quiz 6

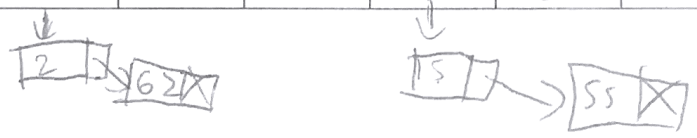
You have 10 minutes to complete the Quiz. The total number of points is 8. You are not allowed to use any books, notes, calculator, or electronic devices. Write your answer carefully and clearly. Incorrect answers will receive little to no points. When you are asked to write a program the program should be clear and include indentations and comments to make it easier to read. Be sure to state any assumptions on which you have based your answers.

Insert next numbers into hash table: **1, 3, 55, 44, 15, 62, 2, 6** by using:

key % size ;

1. Chaining

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
	1	~~62~~	3	44	~~55~~	6			



size = 10

3 % 10 = 3

55 % 10 = 5

44 % 10 = 4

2. Linear probing

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
	1		3						

3. Quadratic probing

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]

4. Double hashing: 11– (value%11)

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]

62 : 2 + (11 - 62 % 11) = 6

1. **Binomial Queue**. Build binomial queue
   4, 1, 3, 2, 16, 9, 10, 14, 8, 7, 12, 25, 30
   When remove max element 3 times