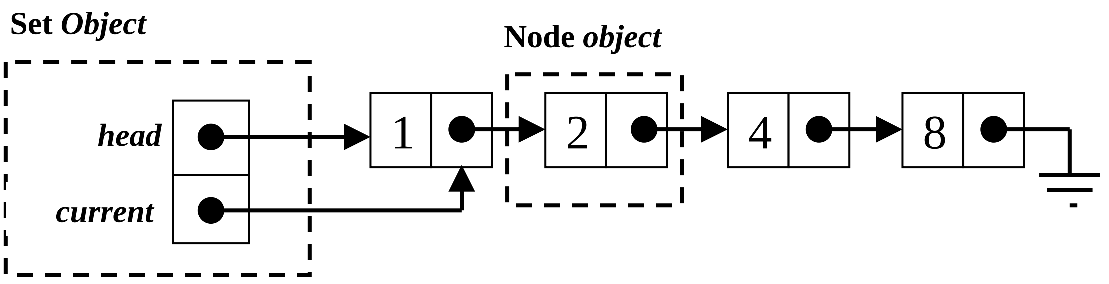
OBJECT ORIENTED PROGRAMMING

1st Midterm Exam

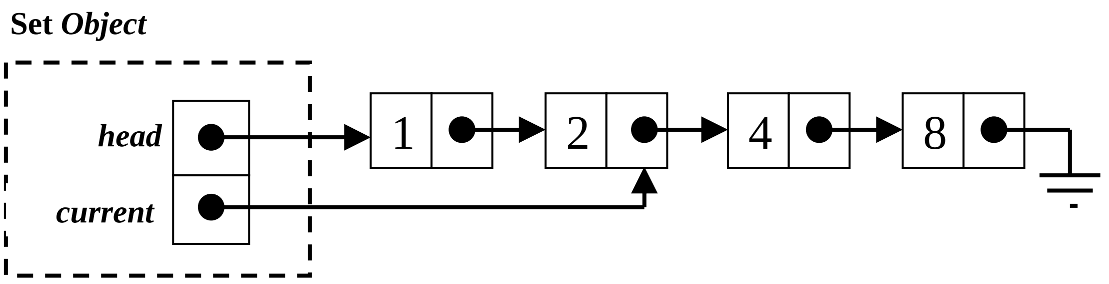
(There are 2 Questions. 2-Hour Exam, Give your answer inside the corresponding box)

Student ID: Name: Signature

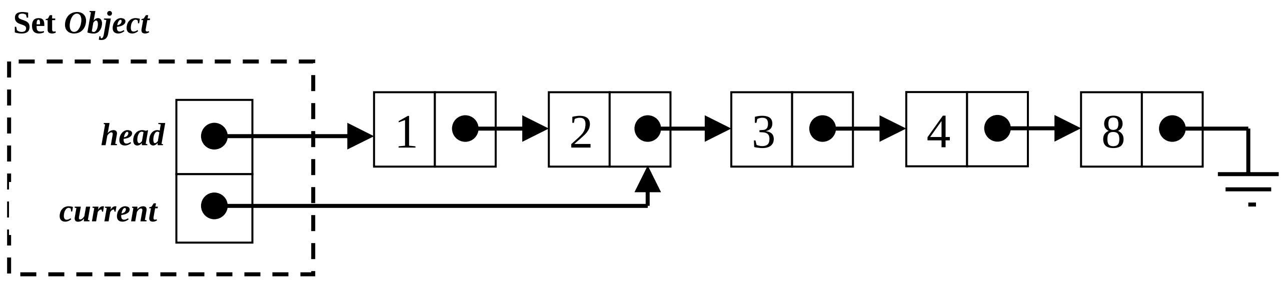
Q.1) (a) (20) You will design a **Set** class in order to model *set* data structure using **object oriented programming** approach. A set contains **unique** elements and keeps its elements **sorted** after insertions and deletions. The following figure will help you to decide on class data members. A typical application code and its output for the sample set (1,2,4,8) is given below.
Initially `head` and `current` points to the first node:



After `set->next()` call, `current` points to the next node:



After `set->add(4)` call the set remains the same since the set already contains “4”.
The call `set->add(3)causes` the set to rearrange its elements so that it remains sorted:



Typical usage of the Set class

```
Set *set = new Set() ;
…
while (set->hasMoreElements())
        cout << endl << "Set element: " << set->next() ;
```

The program output:

```
Set element: 1
Set element: 2
Set element: 3
Set element: 4
Set element: 8
```

```
class Set {
     private:



     public:




} ;
```

Give your answer inside the box

```
class Node {
 private:
    Node * next;
    int hold;
 public:
    Node(int);
    Node* getNextNode()const;
    void setNextNode(Node*);
    int getValue()const;
    void setValue(int);
};
```

Notes and Guidelines:
Consider the **only required** methods.
Do you need default constructor?
Do you need copy constructor?
Do you need destructor?
Do you need assignment operator?

(b) (40) Give the implementations of the methods.

Give your answer inside the box

You may use the back of the page, but not elsewhere

Q.2) (40) What is the output of the following C++ code?

```
class B {
  public:
     B(){ cout << "\nB's constructor"; }
     ~B(){ cout << "\nB's destructor"; }
};
class C {
  private:
     static int n;
  public:
     C(){ cout << "\nC's constructor # " << ++n ; }
     ~C(){ cout << "\nC's destructor # " << n-- ; }
};
int C::n=0;
class A {
  protected:
     B *b;
     C c,d;
  public:
     A(){  b= new B();
           cout << "\nA's constructor"; }
     ~A(){ delete b;
           cout << "\nA's destructor";  }
};
class D : public A {
  public:
     D(){ cout << "\nD's constructor"; }
     ~D(){ cout << "\nD's destructor"; }
};
int main(){
    D a[2];
    return 0;
}
```

Line Number	**Output**
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	
16	
17	
18	
19	
20	