2.1 Prime Number Checker

In [1]:

```
import math
```

In [2]:

```
while True :
    try :
        number = int(input("Enter an integer : "))
        break
    except(ValueError) :
        print("Please enter valid integer value.")
```

```
Enter an integer : 47
```

In [4]:

```
checker = True
for i in range(2,int(math.sqrt(number)+1)) :
    if number % i == 0 :
        print("Entered number is not prime number")
        checker = False
        break
    else :
        continue
if checker :
    print("Entered number is prime number")
```

```
Entered number is prime number
```

2.2 Second Largest from an array

In [11]:

```
Array = []
print("Enter any character to end the input")
while True :
    try :
        element = float(input("Enter element : "))
        Array.append(element)
    except(ValueError) :
        print("You have entered string/character")
        if len(Array) == 0 :
            print("Array is empty please enter some elements")
            continue
        choice = None
        while True :
            try :
                choice = int(input("0.exit\n1.continue\n"))
                break
            except(ValueError) :
                print("Please enter valid option.")
        if choice == 1 :
            continue
        else :
            break
```

```
Enter any character to end the input
Enter element : 89
Enter element : 88
Enter element : 88
Enter element : 87
Enter element : s
You have entered string/character
0.exit
1.continue
0
```

In [12]:

```
Array
```

Out[12]:

```
[89.0, 88.0, 88.0, 87.0]
```

In [13]:

```
UniqueElements = set(Array)
UniqueElements
```

Out[13]:

```
{87.0, 88.0, 89.0}
```

In [14]:

```
largestElement = max(UniqueElements)
largestElement
```

Out[14]:

```
89.0
```

In [15]:

```
UniqueElements.remove(largestElement)
UniqueElements
```

Out[15]:

```
{87.0, 88.0}
```

In [16]:

```
print("Second largest element in the entered list is ",max(UniqueElements))
```

```
Second largest element in the entered list is  88.0
```

2.3 Linked List - Delete Nodes with even values

In [1]:

```
class Node :
    def __init__(self) :
        self.data = None
        self.next = None

class LinkedList(Node) :
    def __init__(self) :
        self.__head = Node()

    def EnterNode(self) :
        if self.__head is None :
            self.__head.data = int(input("Enter node value (head) : "))
            self.__head.next = None
        else :
            node = Node()
            node.data = input("Enter node value : ")
            node.next = None
            tmp = self.__head
            while tmp.next is not None :
                tmp = tmp.next
            tmp.next = node

    def Traverse(self) :
        if self.__head is None :
            print("List is empty")
        else :
            tmp = self.__head.next
            while tmp is not None :
                print(tmp.data,end=" --> ")
                tmp = tmp.next
            print("None")
            del tmp

    def SearchNode(self) :
        if self.__head.next is None :
            print("List is empty cannot search")
        else :
            element = input("Enter the value of the element to be searched : ")
            temp = self.__head.next
            counter = 0
            while (temp is not None) and (temp.data != element):
                print(temp.data)
                counter += 1
                temp = temp.next
            if temp is None :
                print("Element not found")
            else :
                print("Element found at location",counter+1)

    def DeleteNode(self) :
        if self.__head.next is None :
            print("List is already empty")
        else :
            while float(self.__head.next.data) % 2 == 0 :
                self.__head = self.__head.next
                if self.__head.next is None :
                    break
            temp = self.__head.next
            preTemp = temp
```

```
            flag = False
            while (temp is not None) :
                if float(temp.data) % 2 == 0 :
                    if temp.next is not None :
                        preTemp.next = temp.next.next
                    else :
                        preTemp.next = temp.next
                else :
                    preTemp = temp
                if temp is not None and float(temp.data) % 2 == 0:
                    preTemp.next = temp.next
                temp = temp.next
        del temp, preTemp

if __name__ == "__main__" :
    L = LinkedList()
    print("1. Enter Node\n2. Delete Node\n3. Traverse List\n4. Exit")
    Option = int(input("\nEnter option : "))
    while Option != 5 :
        if Option == 1 :
            print()
            L.EnterNode()
            print()
        elif Option == 2 :
            print()
            L.DeleteNode()
            print()
        elif Option == 3 :
            print()
            L.Traverse()
            print()
        elif Option == 4 :
            break
        else :
            print("Please enter correct option")
        print("\n1. Enter Node\n2. Delete Node\n3. Traverse List\n4. Exit")
        Option = int(input("\nEnter option : "))
```

```
1. Enter Node
2. Delete Node
3. Traverse List
4. Exit

Enter option : 1

Enter node value : 65


1. Enter Node
2. Delete Node
3. Traverse List
4. Exit

Enter option : 1

Enter node value : 43


1. Enter Node
2. Delete Node
3. Traverse List
```

```
4. Exit

Enter option : 1

Enter node value : 96


1. Enter Node
2. Delete Node
3. Traverse List
4. Exit

Enter option : 1

Enter node value : 35


1. Enter Node
2. Delete Node
3. Traverse List
4. Exit

Enter option : 1

Enter node value : 47


1. Enter Node
2. Delete Node
3. Traverse List
4. Exit

Enter option : 1

Enter node value : 88


1. Enter Node
2. Delete Node
3. Traverse List
4. Exit

Enter option : 1

Enter node value : 36


1. Enter Node
2. Delete Node
3. Traverse List
4. Exit

Enter option : 1

Enter node value : 21


1. Enter Node
2. Delete Node
3. Traverse List
4. Exit
```

```
Enter option : 1

Enter node value : 45


1. Enter Node
2. Delete Node
3. Traverse List
4. Exit

Enter option : 1

Enter node value : 34


1. Enter Node
2. Delete Node
3. Traverse List
4. Exit

Enter option : 1

Enter node value : 98


1. Enter Node
2. Delete Node
3. Traverse List
4. Exit

Enter option : 1

Enter node value : 33


1. Enter Node
2. Delete Node
3. Traverse List
4. Exit

Enter option : 3

65 --> 43 --> 96 --> 35 --> 47 --> 88 --> 36 --> 21 --> 45 --> 34 --
> 98 --> 33 --> None


1. Enter Node
2. Delete Node
3. Traverse List
4. Exit

Enter option : 2



1. Enter Node
2. Delete Node
3. Traverse List
4. Exit
```

```
Enter option : 3

65 --> 43 --> 35 --> 47 --> 21 --> 45 --> 33 --> None


1. Enter Node
2. Delete Node
3. Traverse List
4. Exit

Enter option : 4
```

2.4 Binary Search Tree - Insert, Search and Delete

In [2]:

```
class Node:
    def __init__(self):
        self.left = None
        self.data = None
        self.right = None
def Insert(root) :
    if root.data is None :
        root.data = int(input("Enter node value : "))
    else :
        newNode = Node()
        newNode.data = int(input("Enter node value : "))
        while root.data <= newNode.data :
            if root.right :
                root = root.right
            else :
                break
        while root.data > newNode.data :
            if root.left :
                root = root.left
            else :
                break
        if newNode.data > root.data :
            root.right = newNode
        else :
            root.left = newNode

def Search(root,element,level) :
    if root :
        if root.data == element :
            print("\nElement is found at level",level,"\n")
        elif root.data < element :
            Search(root.right,element,level+1)
        elif root.data > element :
            Search(root.left,element,level+1)
        else :
            print("\nSorry !! Entered element is not in the Binary Search Tree!!!\n

#Minimum from the right sub tree
def inOrderSuccessor(root) :
    temp = root
    while temp.left :
        temp = temp.left
    return temp

def Delete(root,element) :
    if root is None :
        return root
    if element < root.data :
        root.left = Delete(root.left,element)
    elif element > root.data :
        root.right = Delete(root.right,element)
    else :
        if root.left is None :
            temp = root.right
            root = None
            return temp
        elif root.right is None :
            temp = root.left
            root = None
```

```
            return temp
        temp = inOrderSuccessor(root.right)
        root.data = temp.data
        root.right = Delete(root.right,temp.data)
    return root

def inOrderTraversal(temp) :
    if temp :
        inOrderTraversal(temp.left)
        print(temp.data,end="\t")
        inOrderTraversal(temp.right)

def preOrderTraversal(temp) :
    if temp :
        print(temp.data,end="\t")
        preOrderTraversal(temp.left)
        preOrderTraversal(temp.right)

def postOrderTraversal(temp) :
    if temp :
        postOrderTraversal(temp.left)
        postOrderTraversal(temp.right)
        print(temp.data,end="\t")


Root = Node()
Choice = int(input("\n\n1. Insert\n2. Search\n3. Delete\n4. Traverse\n5. Exit\n\nEn
while Choice != 5 :
    if Choice == 1 :
        Insert(Root)
    elif Choice == 2 :
        element = int(input("\nEnter the value to search : "))
        Search(Root,element,0)
    elif Choice == 3 :
        element = int(input("\nEnter an element to delete : "))
        Root = Delete(Root,element)
    elif Choice == 4 :
        print("Inorder Traversal")
        inOrderTraversal(Root)
        print("\nPreorder Traversal")
        preOrderTraversal(Root)
        print("\nPostorder Traversal")
        postOrderTraversal(Root)
    Choice = int(input("\n\n1. Insert\n2. Search\n3. Delete\n4. Traverse\n5. Exit\n
```

```
1. Insert
2. Search
3. Delete
4. Traverse
5. Exit

Enter Choice : 1
Enter node value : 30


1. Insert
2. Search
3. Delete
4. Traverse
```

```
5. Exit

Enter Choice : 1
Enter node value : 40


1. Insert
2. Search
3. Delete
4. Traverse
5. Exit

Enter Choice : 1
Enter node value : 80


1. Insert
2. Search
3. Delete
4. Traverse
5. Exit

Enter Choice : 1
Enter node value : 60


1. Insert
2. Search
3. Delete
4. Traverse
5. Exit

Enter Choice : 1
Enter node value : 20


1. Insert
2. Search
3. Delete
4. Traverse
5. Exit

Enter Choice : 1
Enter node value : 10


1. Insert
2. Search
3. Delete
4. Traverse
5. Exit

Enter Choice : 4
Inorder Traversal
10      20      30      40      60      80
Preorder Traversal
30      20      10      40      80      60
Postorder Traversal
10      20      60      80      40      30

1. Insert
```

```
2. Search
3. Delete
4. Traverse
5. Exit

Enter Choice : 2

Enter the value to search : 30

Element is found at level 0



1. Insert
2. Search
3. Delete
4. Traverse
5. Exit

Enter Choice : 3

Enter an element to delete : 30


1. Insert
2. Search
3. Delete
4. Traverse
5. Exit

Enter Choice : 4
Inorder Traversal
10      20      40      60      80
Preorder Traversal
40      20      10      80      60
Postorder Traversal
10      20      60      80      40

1. Insert
2. Search
3. Delete
4. Traverse
5. Exit

Enter Choice : 3

Enter an element to delete : 10


1. Insert
2. Search
3. Delete
4. Traverse
5. Exit

Enter Choice : 4
Inorder Traversal
20      40      60      80
Preorder Traversal
40      20      80      60
Postorder Traversal
```

```
20      60      80      40

1. Insert
2. Search
3. Delete
4. Traverse
5. Exit

Enter Choice : 5
```

In [ ]: