* Set

- A set is a container that stores a collection of unique values over a given comparable domain in which the stored values have no particular ordering.

→ set():

- Creates a new set initialized to the empty set.

→ Length():

- Returns the number of elements in the set, also known as the cardinality. Accessed using the len() function.

→ contains(element):

- Determines if the value is an element of the set and returns the appropriate boolean value.
- Accessed using the in operator.

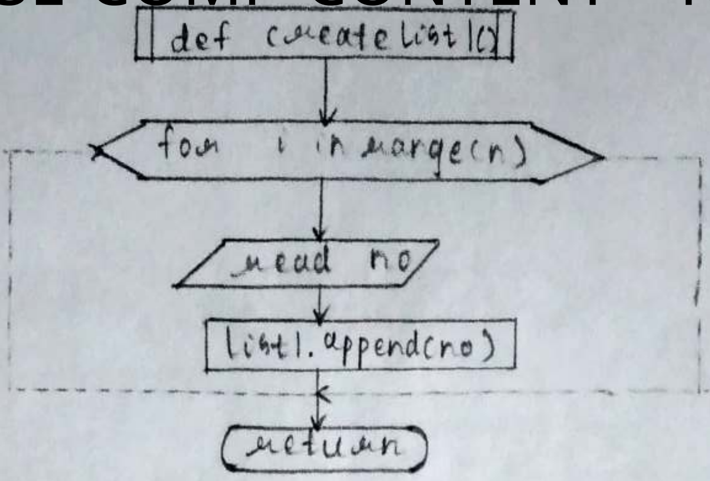
→ add(element):

- Modifies the set by adding the given value of an element to the set if the element is not already a member.
- If the element is not unique, no action is taken and the operation is skipped.

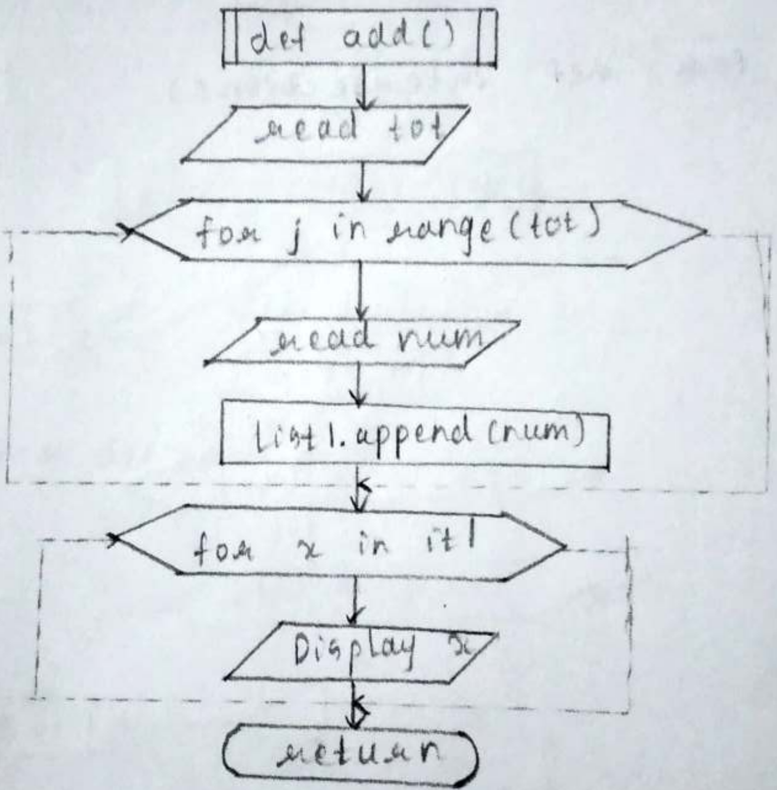
→ remove(element):

- Removes the given value from the set if the value is contained in the set and raises an exception otherwise.

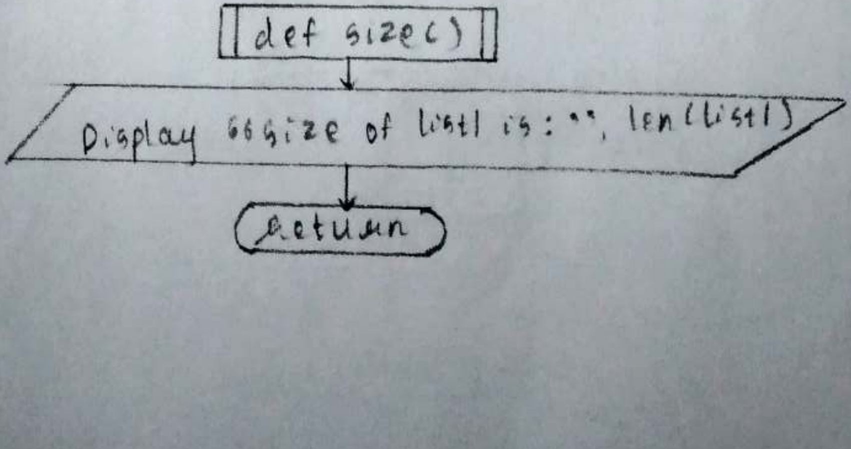
wchart for createlist1()



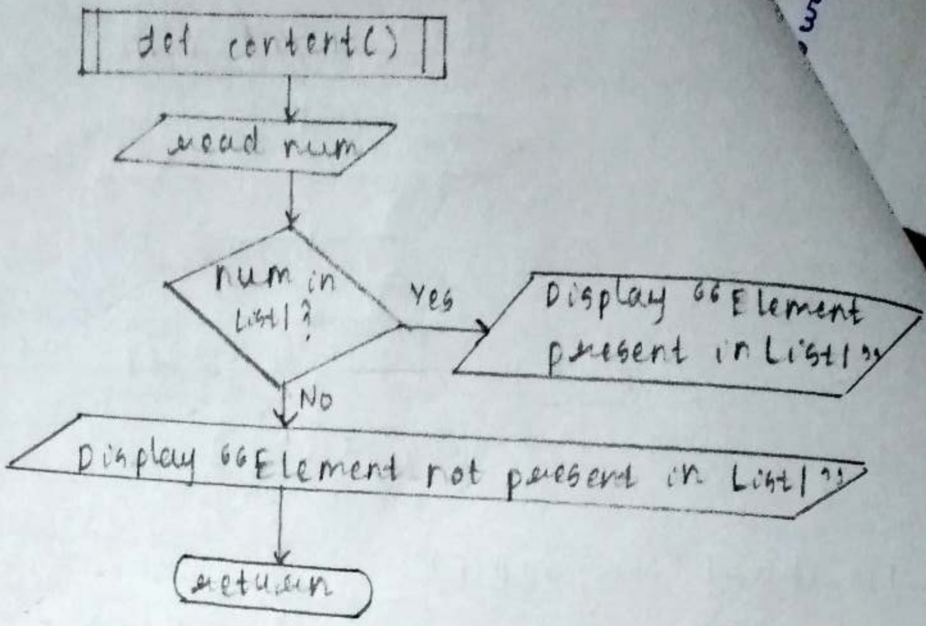
→ Flowchart for add()



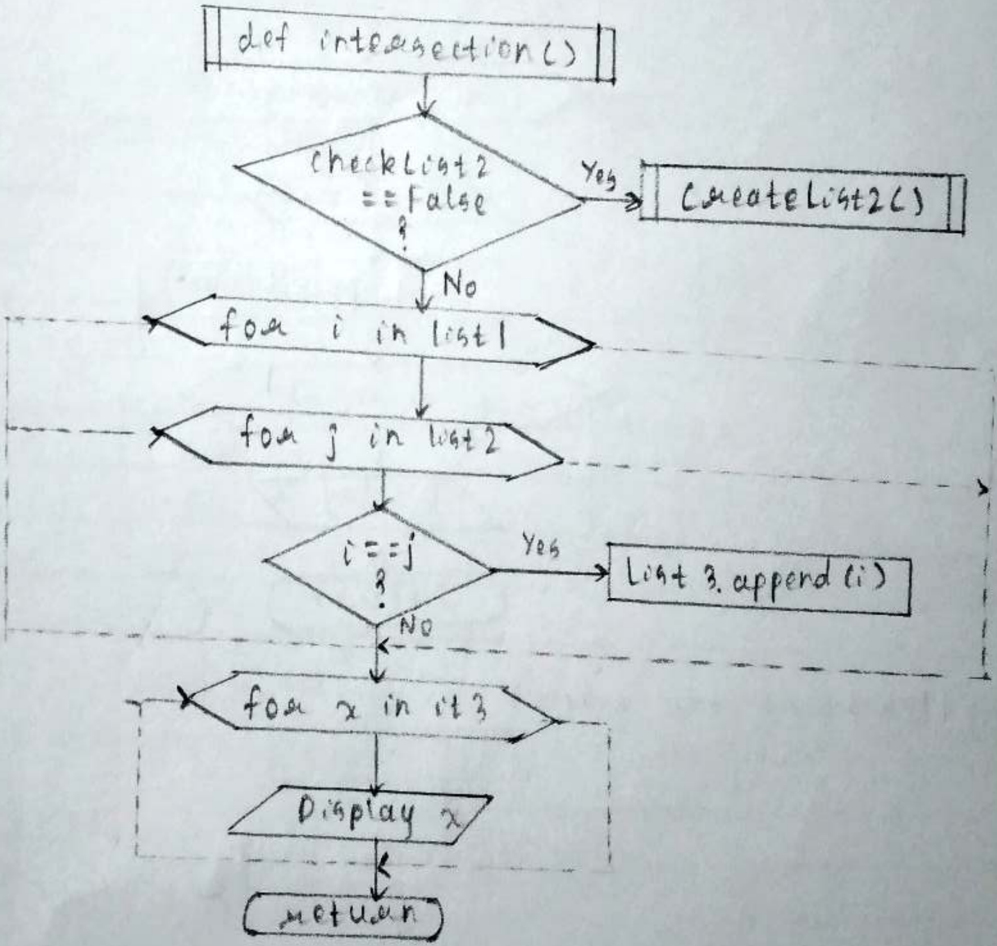
→ Flowchart for size()



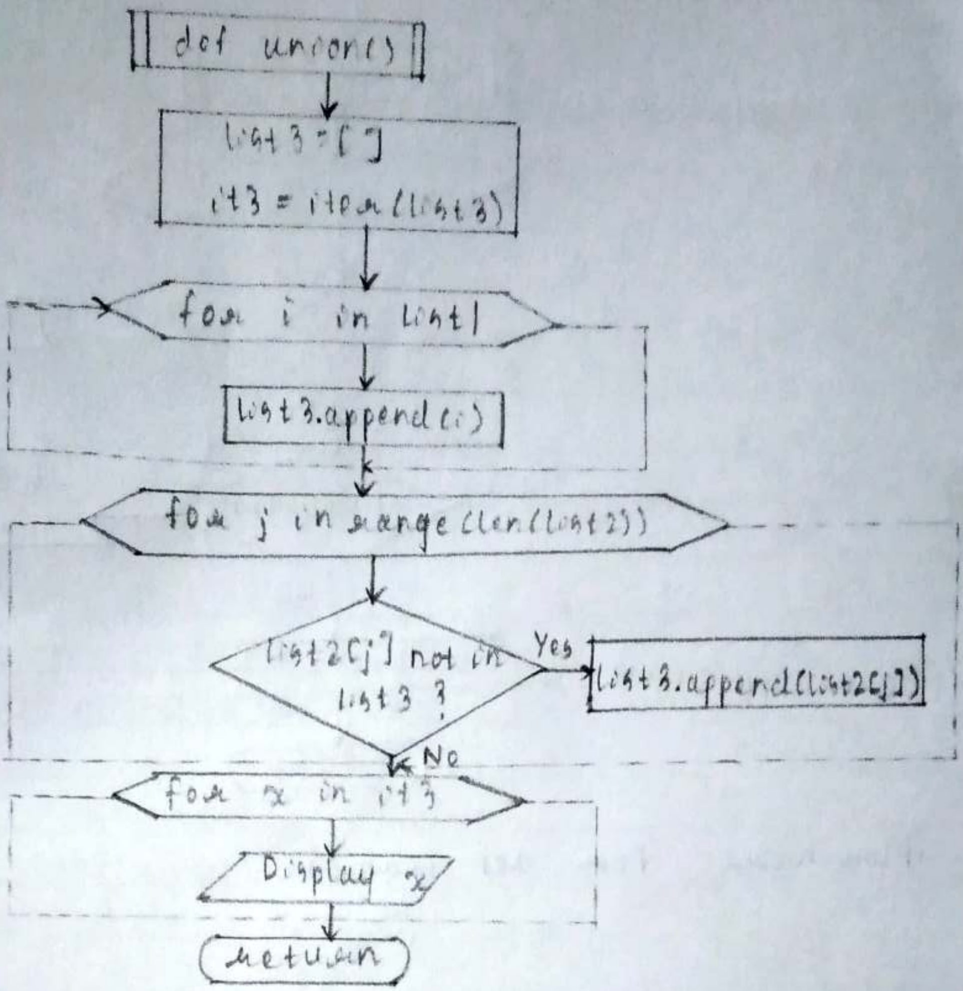
→ Flowchart for def content()


def content()
read num
num in List1 ?
Yes
Display "Element present in List1"
No
Display "Element not present in List1"
return

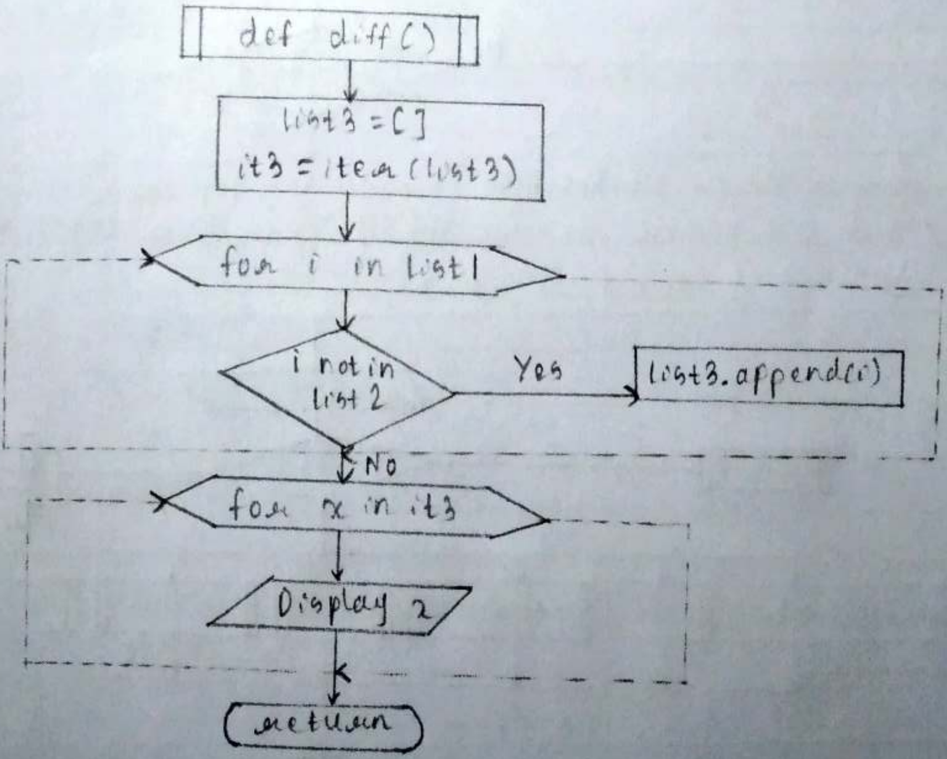
→ Flowchart for def intersection()



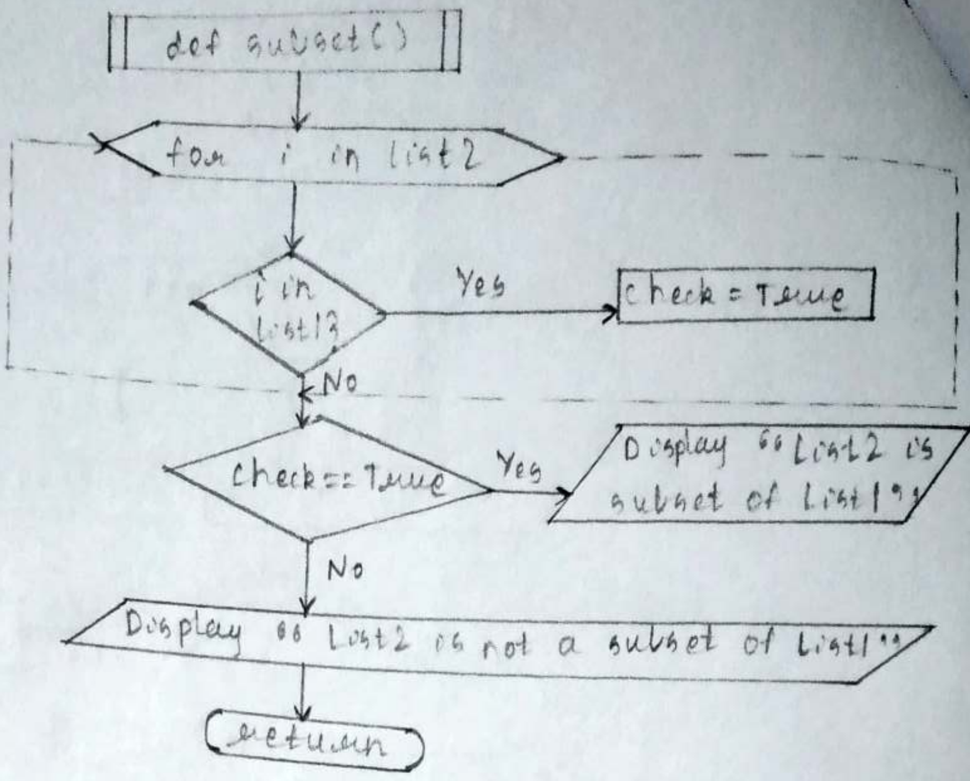
flowchart for def union()



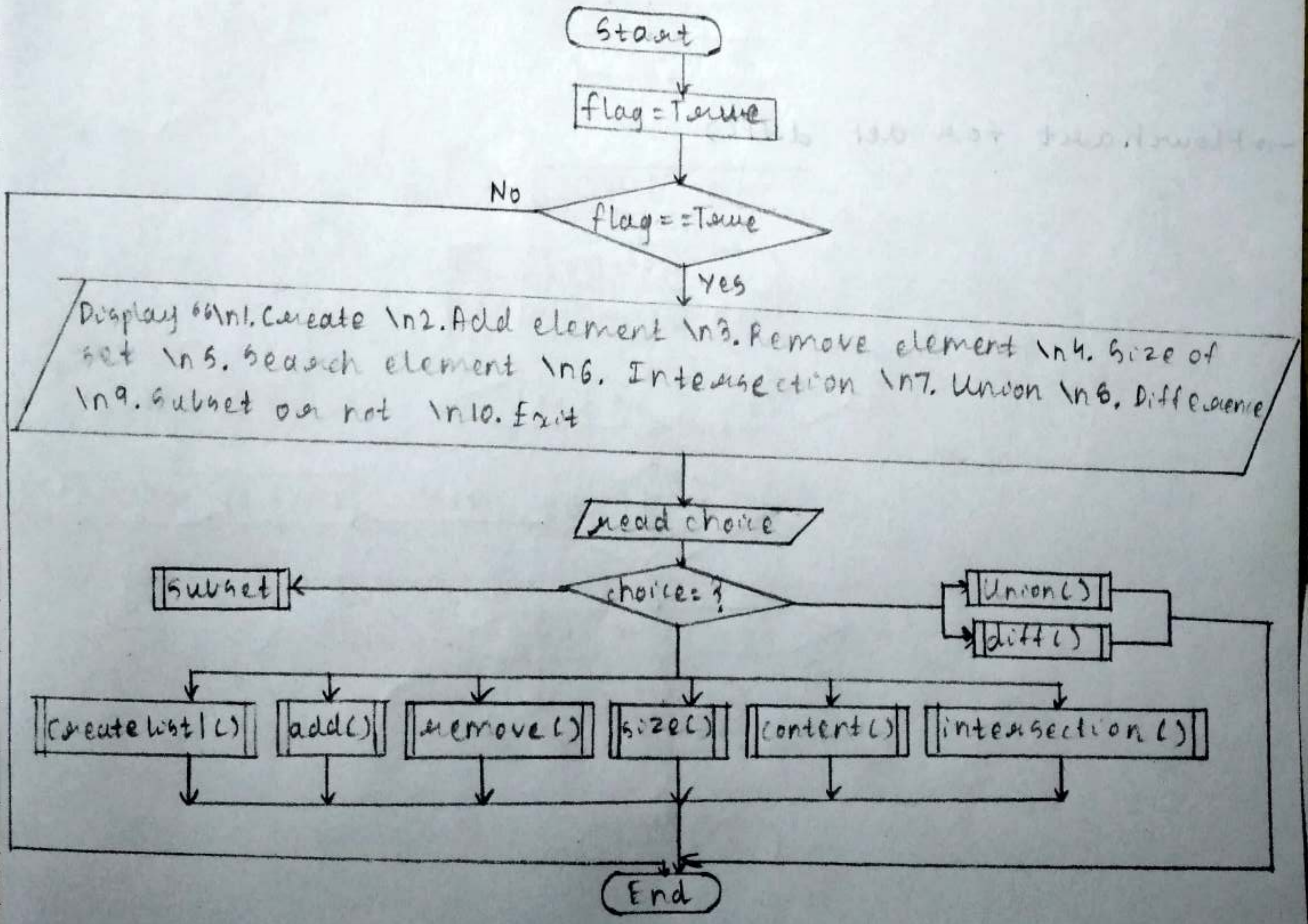
→ Flowchart for def diff()



→ ~~de~~ Flowchart for def subset()



→Flowchart for def main()


Start
flag=True
flag==True
No
Yes
Display "\n1. Create \n2. Add element \n3. Remove element \n4. Size of set \n5. Search element \n6. Intersection \n7. Union \n8. Difference \n9. Subset or not \n10. Exit
read choice
choice=?
subset
union()
diff()
createlist1()
add()
remove()
size()
contect()
intersection()
End

→ Pseudocode for createlist1():

```
1. for i in range(n) do
   begin
           read no
           list1.append(no)
   end
2. return
```

→ Pseudocode for add():

```
1. read tot
2. for j in range(tot) do
   begin
           read num
           list1.append(num)
   end
3. for x in it1: do
   begin
           Display x
   end
4. return
```

→ Pseudocode for remove():

```
1. Read rem
2. Declare check=False
3. for i in List1 do
   begin
        if rem==i then
             list1.remove(rem)
             for x in it1 do
             begin
                   Display x
             end
```

```
            Declare check = True
   end
4. if check == False then
        Display "Element not in List"
5. return

→ Pseudocode for size()
1. Display "Size of List1 is: ", len(List1)
2. return

→ Pseudocode for content()
1. Read num
2. if num in list1 then
        Display "Element present in List1"
   else
        Display "Element not present in List1"
3. return

→ Pseudocode for intersection()
1. Declare List3 = [ ]
2. Store iter(List3) in it3
3. if checkList2 == False then
        call function createList2 ()
4. for i in List1 do
   begin
        for j in List2 do
        begin
              if i == j then
                    List3.append (i)
        end
   end
5. for x in it3 do
```

```
begin
        Display x
end
6. return
```

→ Pseudo code for union()

```
1. Declare list3 = []
2. store iter(list3) in it3
3. if checkList2 == False then
        call function createList2()
4. for i in list1 do
   begin
        list3.append(i)
   end
5. for j in range(len(list2)) do
   begin
        if list2[j] not in list3 then
                list3.append(list2[j])
   end
6. for x in it3 do
   begin
        Display x
   end
7. return
```

→ Pseudo code for diff()

```
1. Declare list3 = []
2. store iter(list3) in it3
3. if checkList2 == False then
        Create List2()
4. for i in list1 do
   begin
```

```
            if i not in List2 then
                    List3. append(i)
   end
5. for x in it3 do
   begin
           Display x
   end
6. return
```

→ Pseudocode for subset()

```
1. Declare check = False
2. if checkList2 == False then
           call function createList2()
3. for i in List2 do
   begin
          if i in list1 then
                  Declare check = True
   end
4. if check == True then
          Display "List2 is subset of List1"
5. else
          Display "List2 is not a subset of List1"
```

→ Pseudocode for main()

```
1. Start
2. Declare flag = True
3 while flag == True do
  begin
        Display "\n1. Create set \n2 Add Element \n3. Remove
                 element \n3. size of set \n5. Search
                 element \n6. Intersection \n7. Union
                 \n8. Difference \n9. Subset or not
                 \n10. Exit"
```

```
      read choice
      if choice ==1 then
            call function createlist1()
      elif choice ==2 then
            call function add()
      elif choice == 3 then
            call function remove()
      elif choice == 4 then
            call function size
      elif choice ==5 then
            call function content ()
      elif choice ==6 then
            call function intersection ()
      elif choice ==7 then
            call function union()
      elif choice ==8 then
            call function diff()
      elif choice == 9 then
            call function subset()
      elif choice ==10 then
            ~~call~~ set flag= False

   end
4- stop
```

Q.1 Define ADT of SET.

Ans. • Sets are a type of abstract data type that allows you to store a list of non-repeated values.

• Their name derives from the mathematical concept of finite sets.

1) Set():
• Creates a new set initialized to the empty set.

2) length():
• Returns the number of elements in the set, also known as the cardinality.
• Accessed using the len() function.

3) contains (element):
• Determines if the given value is an element of the set and returns the appropriate boolean value.
• Accessed using the in operator.

4) add (element):
• Modifies the set by adding the given value or element to the set if the element is not already a member.
• If the element is not unique, no action is taken and operation is skipped.

5) Remove (element):
• Removes the given value from the set if the value is contained ~~in~~ in the set and raises an exception otherwise

Q2. Explain iterator in C++ STL ?.

Ans. • The concept of an iterator is fundamental to understanding the C++ Standard Template Library (STL) because iterators provide a means for accessing data stored in container classes such as vector, map, list, etc.

- All containers support a function called begin, which will return an iterator pointing to the beginning of the container (the first element) and function, end, that returns an iterator corresponding to having reached the end of the container.
- In fact, you can access the element by "dereferencing" the iterator with a *, just as you would dereference a pointer.

→ Syntax:

~~St~~ class_name <template-parameters>::iterator name