

## \* 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 on 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.

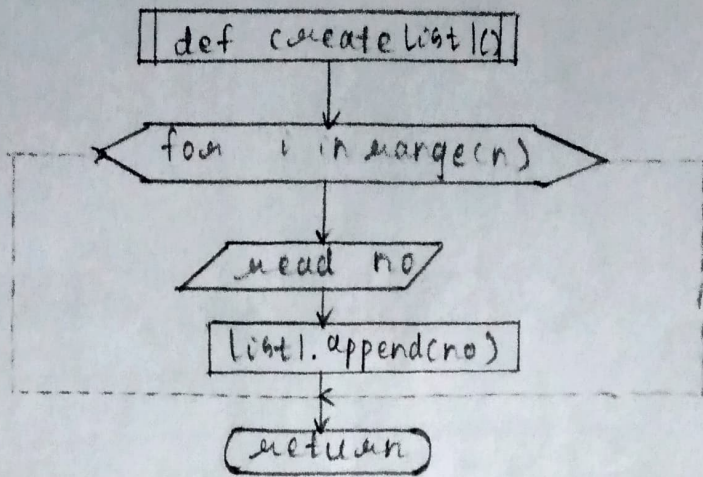
→ `add(element)`:

- 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.

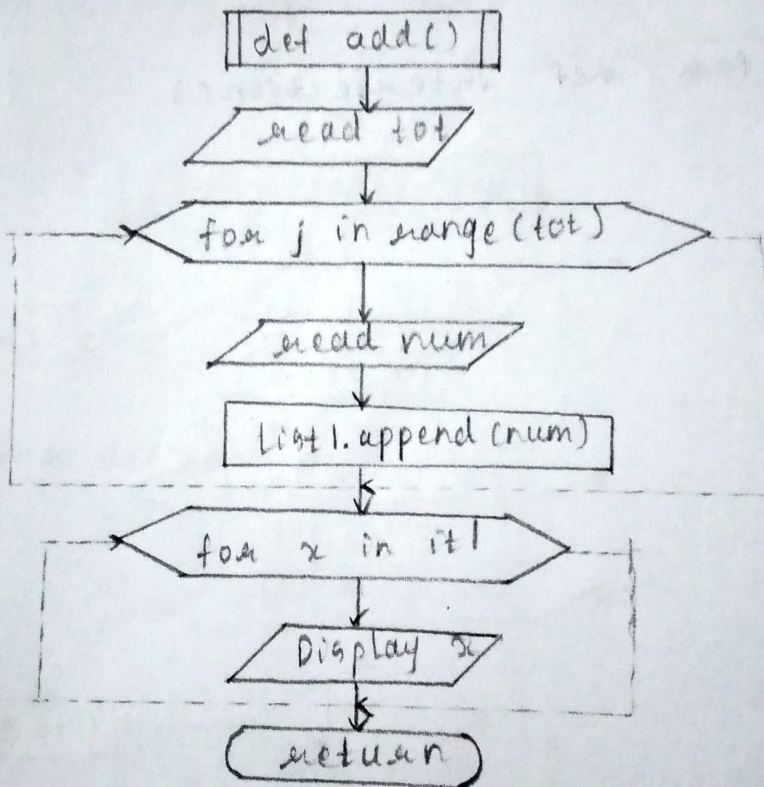
→ `remove(element)`:

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

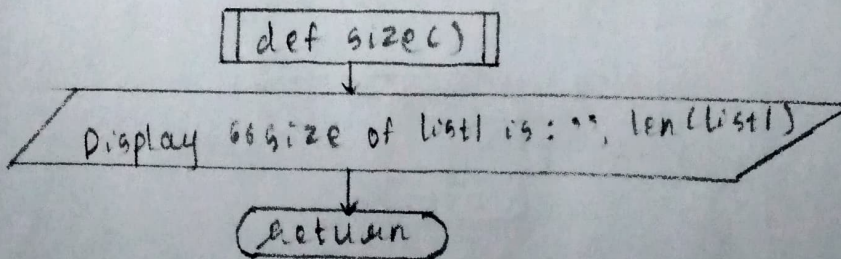
Flowchart for createList()



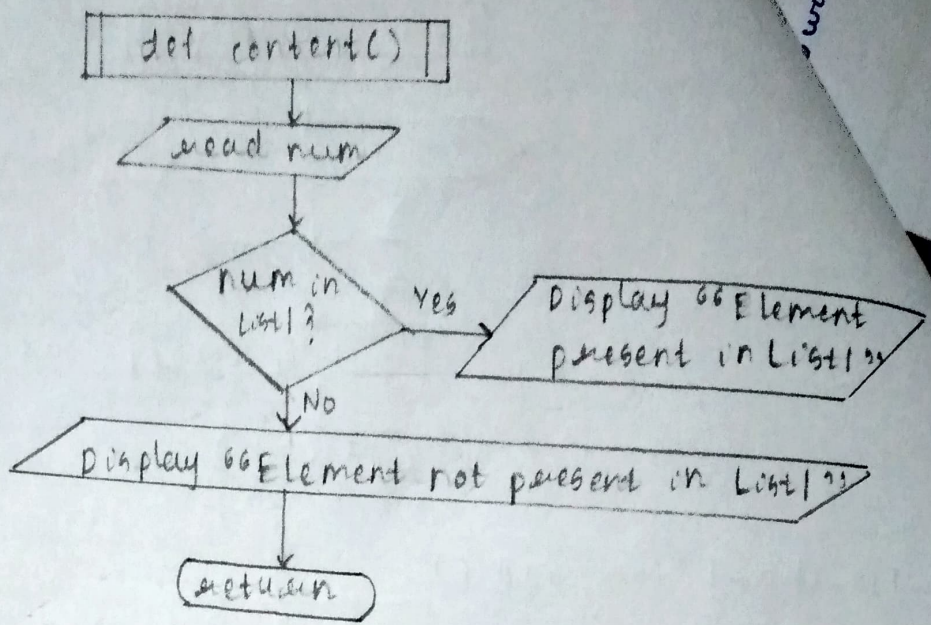
→ Flowchart for add()



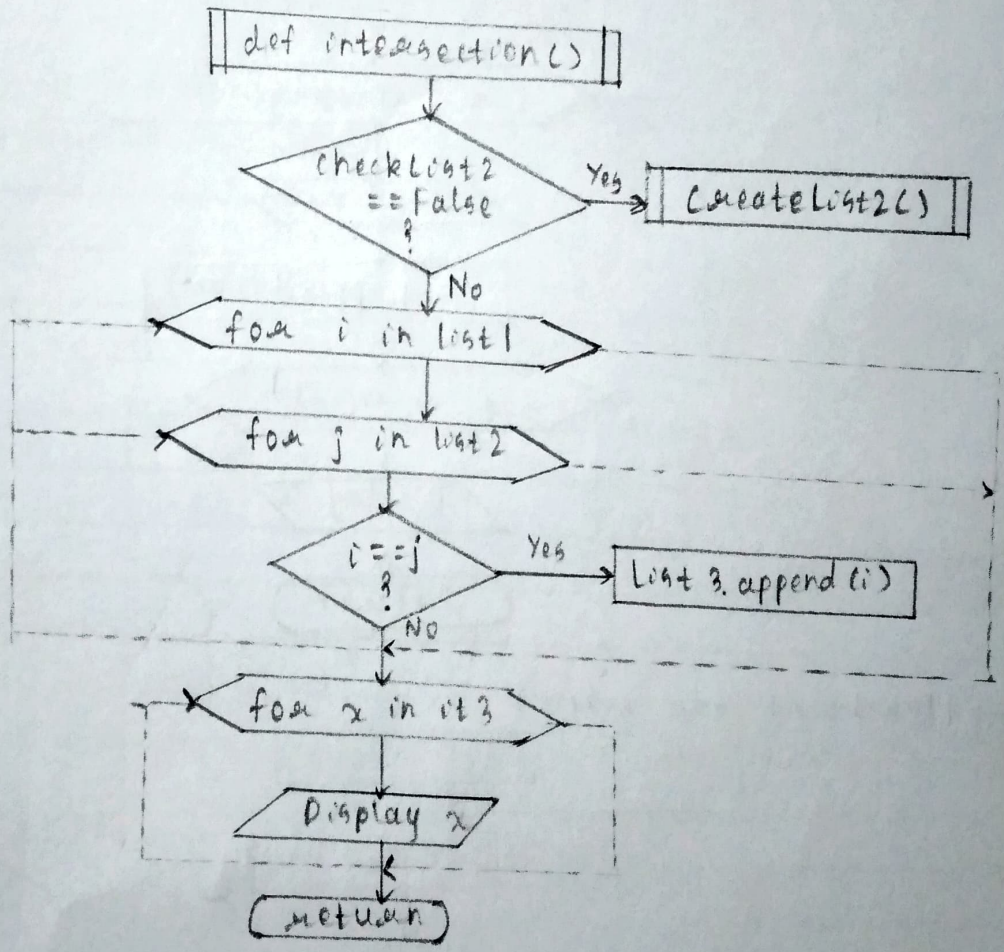
→ Flowchart for size()



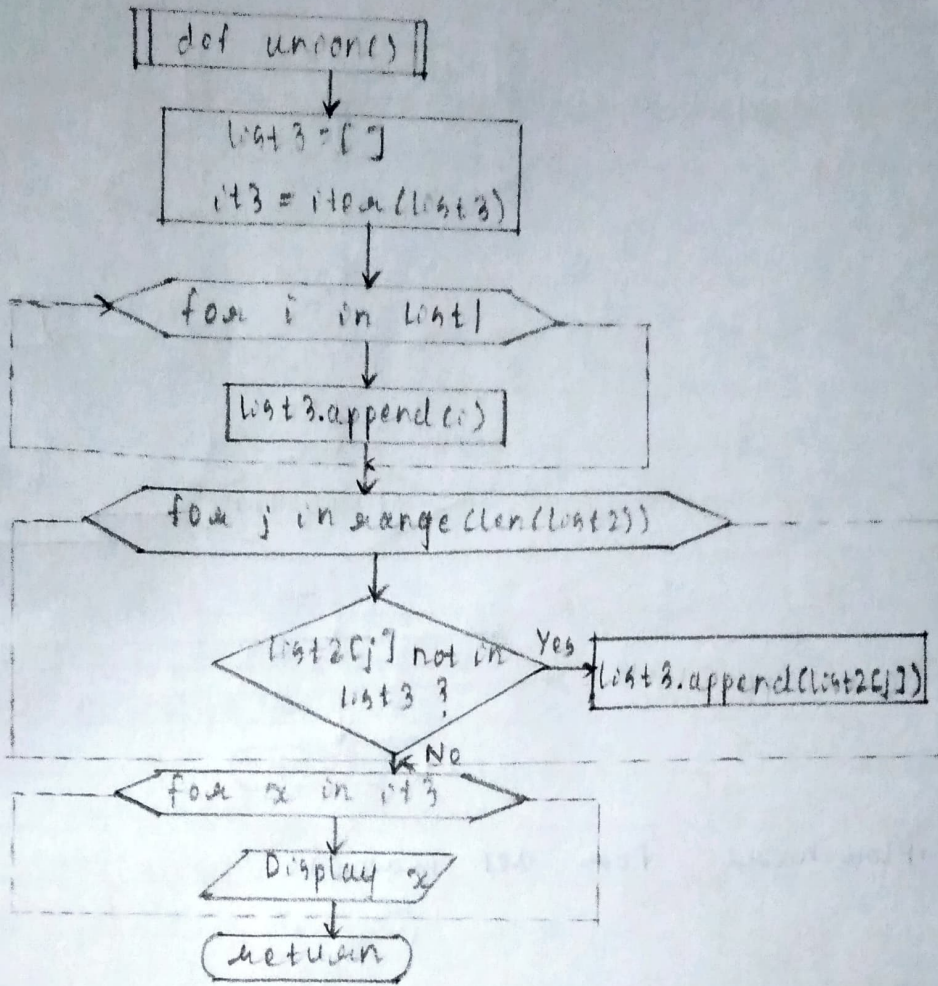
→ Flowchart for def content (C)



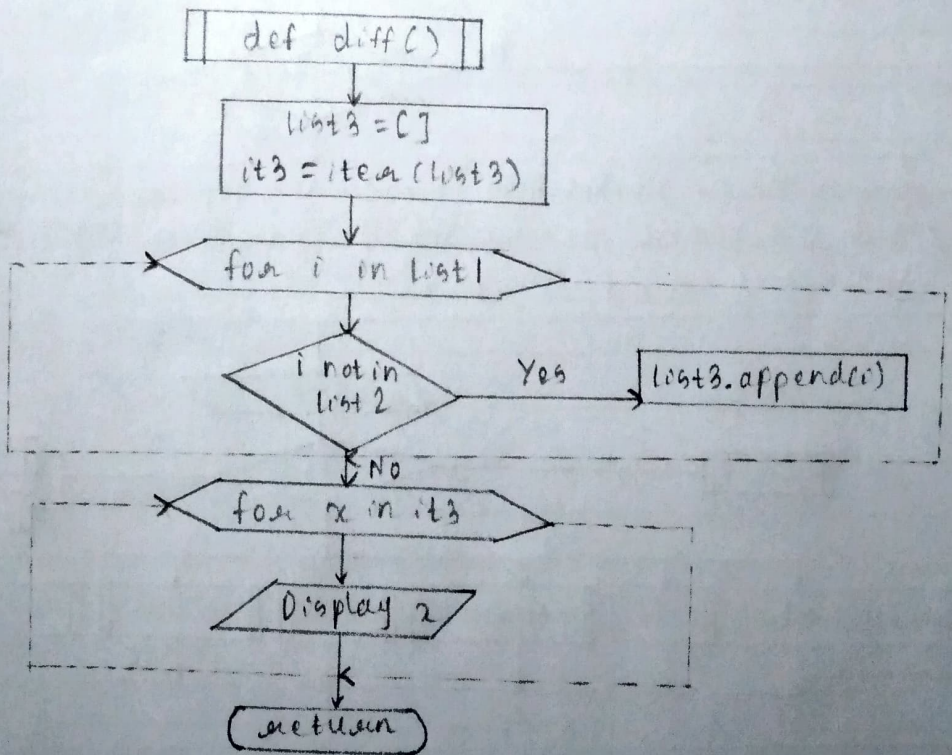
→ Flowchart for def intersection (C)



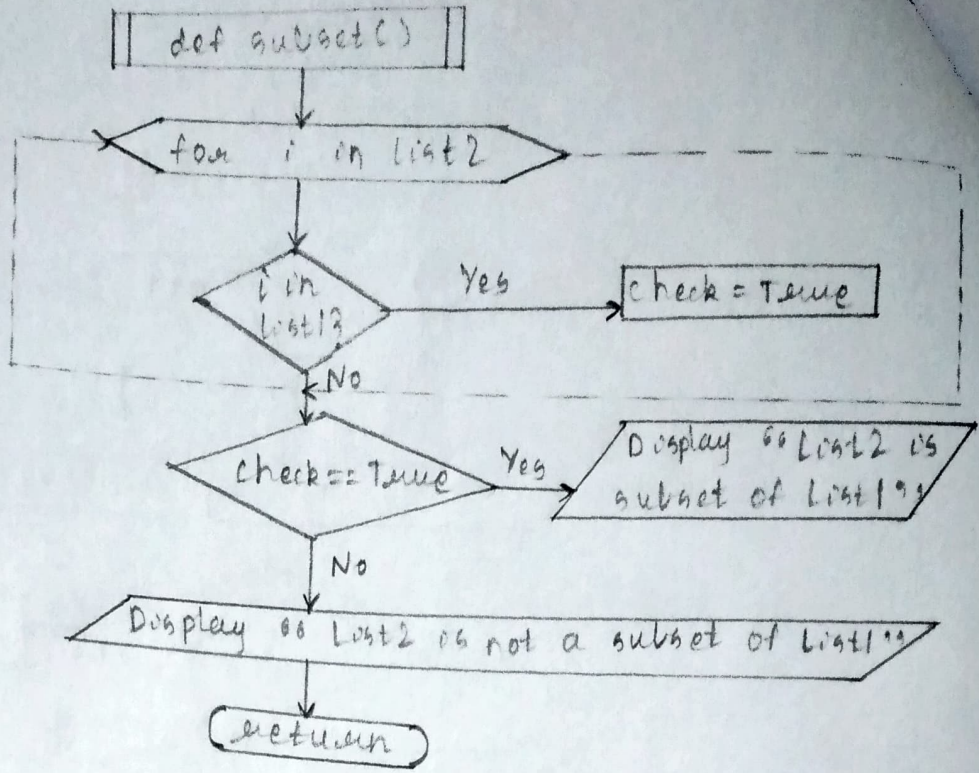
Flowchart for def unique()



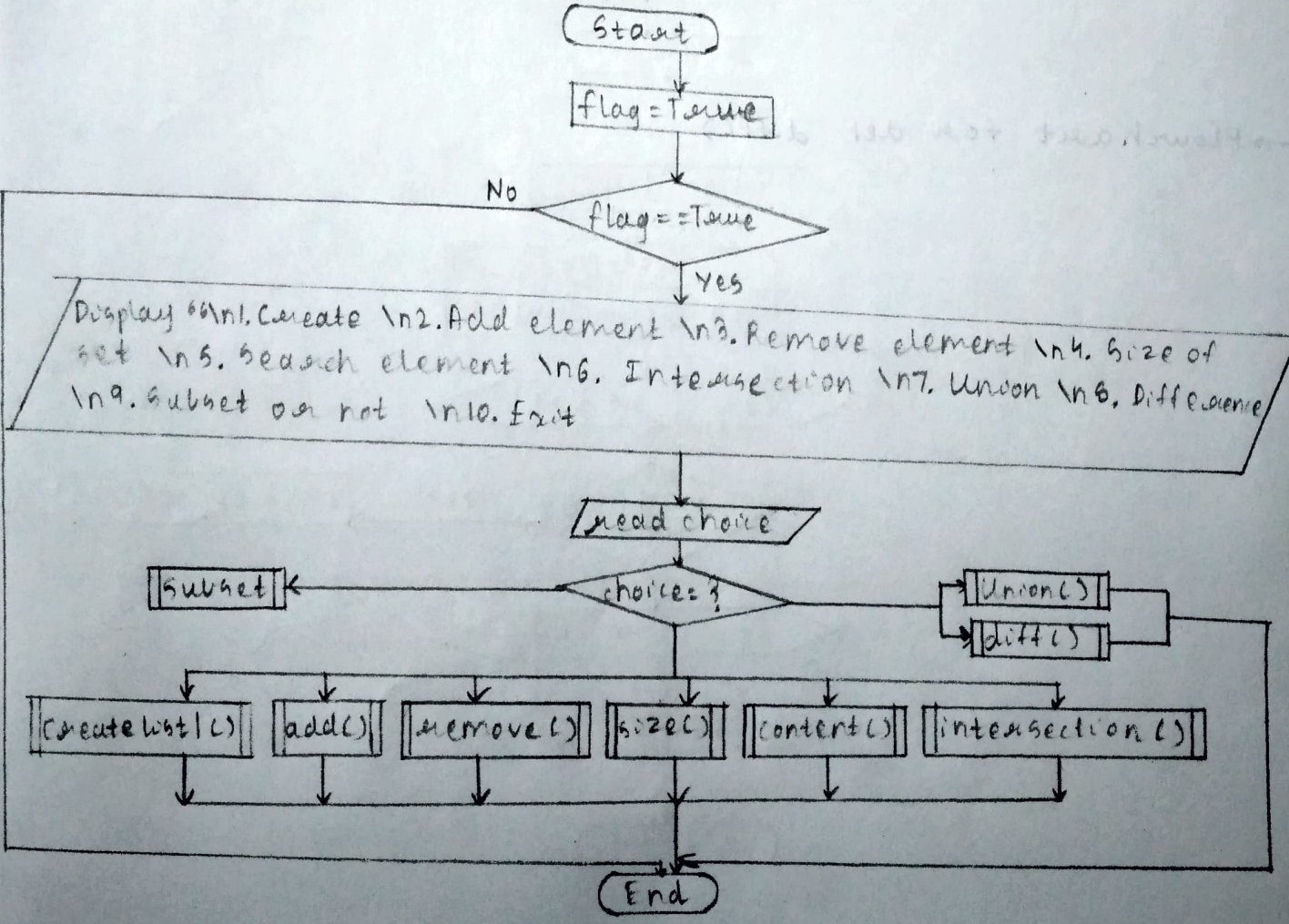
→ Flowchart for def diff()



→ Flowchart for def subset()



→ Flowchart for def main()



→ Pseudocode for createList():

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 list1: 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 list1 do  
begin  
    Display  $x$   
end  
end

Declare check = True

end

4. if check == False then

Display "Element not in List"

5. return

→ Pseudocode for size()

1. Display "size of List is: ", len(list)

2. return

→ Pseudocode for content()

1. Read num

2. if num in list then

Display "Element present in List"

else

Display "Element not present in List"

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

→ Pseudocode for union()

1. Declare list3 = []

2. store iter(list3) in it3

3. if check list2 == 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

→ Pseudocode for diff()

1. Declare list3 = []

2. store iter(list3) in it3

3. if check list2 == 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 list3 do
```

```
begin
```

```
    display x
```

```
end
```

```
6. return
```

→ Pseudocode for subset C)

```
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 "1. Create set 2. Add Element 3. Remove  
element 4. size of set 5. Search  
element 6. Intersection 7. Union  
8. Difference 9. Subset or not  
10. Exit"
```

```
read choice
if choice == 1 then
    call function createList()
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

## Define ADT of SET.

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 as 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 on 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:

```
class class_name <template-parameters>; iterator name
```