PROBLEM STATEMENT

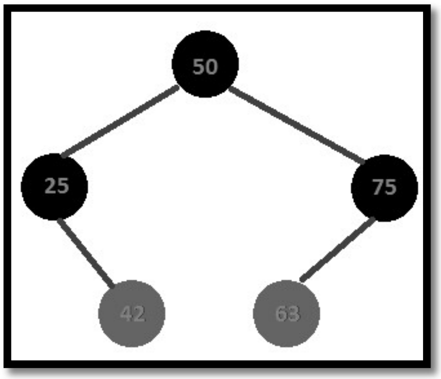
To provide a generic implementation of the set data structure in the C programming language using red-black trees.

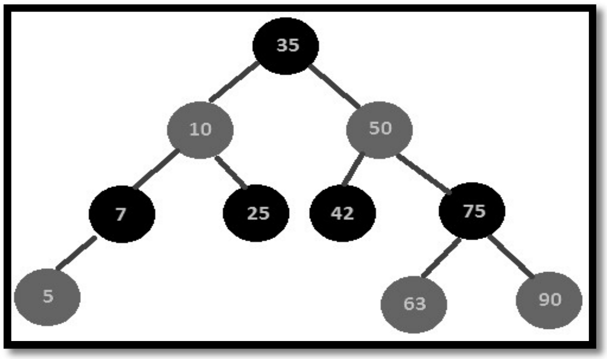
DESIGN

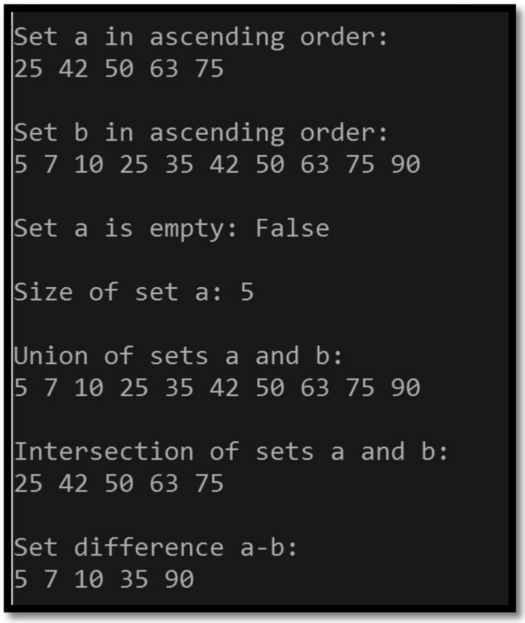
- We have implemented methods for sets using insertion, deletion, traversal and searching in red black trees.
- A set is represented as a structure containing a pointer to the root of the red-black tree, a comparator, a function to print the data and a pointer to the external/ failure node.
- A set is composed of one or more nodes of a red-black tree. Each node has a colour, data, and pointers to its left child, its right child and its parent.

TEST CASES

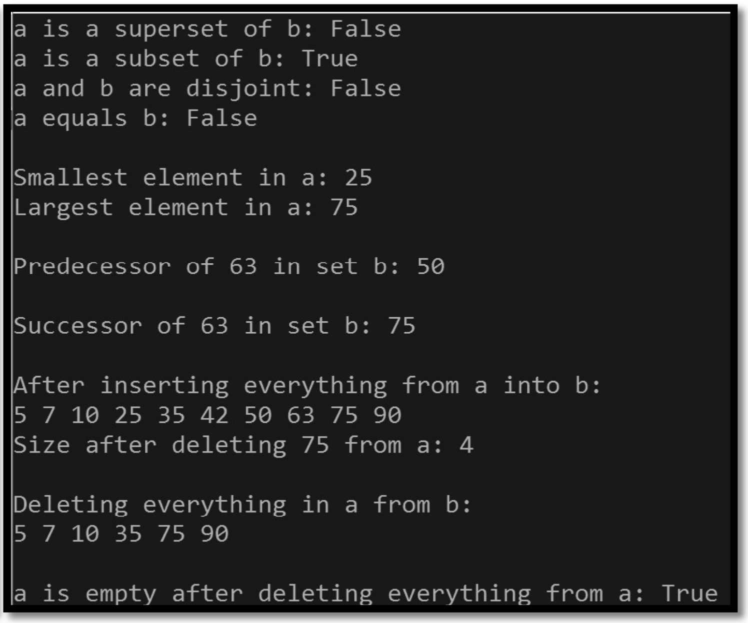
Sets of integer type:
Set a: 25, 42, 50, 63, 75
Set b: 5, 7, 10, 25, 35, 42, 50, 63, 75, 90





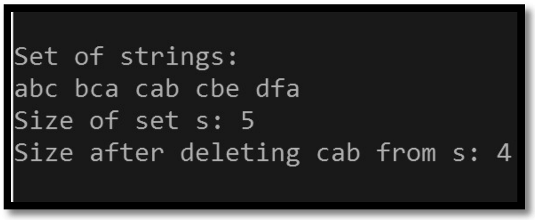


```
Set a in ascending order:
25 42 50 63 75

Set b in ascending order:
5 7 10 25 35 42 50 63 75 90

Set a is empty: False

Size of set a: 5

Union of sets a and b:
5 7 10 25 35 42 50 63 75 90

Intersection of sets a and b:
25 42 50 63 75

Set difference a-b:
5 7 10 35 90
```



```
a is a superset of b: False
a is a subset of b: True
a and b are disjoint: False
a equals b: False

Smallest element in a: 25
Largest element in a: 75

Predecessor of 63 in set b: 50

Successor of 63 in set b: 75

After inserting everything from a into b:
5 7 10 25 35 42 50 63 75 90
Size after deleting 75 from a: 4

Deleting everything in a from b:
5 7 10 35 75 90

a is empty after deleting everything from a: True
```

Set of strings s: "abc", "bca", "cab", "dfa", "cbe"



```
Set of strings:
abc bca cab cbe dfa
Size of set s: 5
Size after deleting cab from s: 4
```

INSTALLATION

- Copy the file “set.h”, included in the CD into a folder of your choice.
- If your program is in the same folder as “set.h” ,insert the statement, #include “set.h” in your program.
- Otherwise mention the entire path of “set.h”.

USER MANUAL

CREATING A SET OF A DATA TYPE FOR THE FIRST TIME IN A PROGRAM:

Include the following statements in your code (outside a function):

- set_declare(type);
- set_define(type);

For example:

- set_declare(int);
- set_define(int);

If the data type includes spaces or *, typedef will have to be used.
For example:

- typedef char* string;
- set_declare(string);
- set_define(string);

OPERATIONS:

- Create set
- Insert, Insert all
- Delete, Delete all, Clear
- Search
- IsEmpty, Size
- Union, Intersection, Difference
- Subset, superset, disjoint, equals
- Minimum, Maximum, Predecessor, Successor
- Inorder, ToArray

LIST OF FUNCTIONS, PARAMETERS AND THEIR RETURN TYPES:

Function Name	Parameters	Return Type	Description
create_set_type	int (*comparator)(const type p1, const type p2), void (*print_data)(const type data)	set_type*	Takes functions to compare objects of the data type and print objects as arguments, and returns a set of that type.
insert_type	set_type *set, type element	void	Inserts the element of the specified type into the set. It has no effect if the element is already present.
delete_type	set_type *set, type element	void	Deletes the element from the set. It has no effect if the element is not present.
search_type	set_type *set, type element	int	Returns 1 if the element is present in the set, -1 otherwise.
isEmpty_type	set_type* set	int	Returns 1 if the set is empty, -1 otherwise.
size_type	set_type* set	long unsigned int	Returns the number of elements in the set. Returns 0 if the set is empty.
union_type	set_type* a, set_type* b	set_type*	Returns the union of sets a & b.
intersection_type	set_type* a, set_type* b	set_type*	Returns the intersection of sets a & b.
difference_type	set_type* a, set_type* b	set_type*	Returns a-b.
min_type	set_type *set	type	Returns the smallest element in the set.
max_type	set_type *set	type	Returns the largest element in the set.
issuperset_type	set_type* a, set_type* b	int	Returns 1 if a is a superset of b, -1 otherwise.
issubset_type	set_type* a, set_type* b	int	Returns 1 if a is a subset of b, -1 otherwise.
isdisjoint_type	set_type* a, set_type* b	int	Returns 1 if a and b are disjoint, -1 otherwise.
equals_type	set_type* a, set_type* b	int	Returns 1 if a and b are equal, -1 otherwise.
toArray_type	set_type* a	type*	Returns an array containing the elements of the set.
insertall_type	set_type* a,set_type* res	void	Inserts all the elements of a

			into res.
deleteall_type	set_type* a, set_type* res	void	Deletes all the elements of a from res.
clear_type	set_type* set	void	Deletes all the elements in the set.
successor_type	set_type *set, type element	type	Returns the smallest element in the set, greater than the given element.
predecessor_type	set_type *set, type element	type	Returns the largest element in the set, smaller than the given element.
inorder_type	set_type *set	void	Displays the elements of the set in ascending order.