Data Structures used:-

```
/*------Node--------*/

struct node {
	char node_tkn[100];		//node's name  ----- also called field one
	char node_tag[100];		//node's tag   ----- also called field two
	GList *childs;			//a list containing node's children
	int child_count;		//number of children i.e size of the list childs
	struct node *parent;		//parent of this node
	or_node *OR;
} ;
typedef struct node node;


/*-------OR Node -------*/
struct or_node {
	int fs_count;
	fs_struct *fs[5];
};
typedef struct or_node or_node;


/*--------feature structure---------*/
struct fs_struct {
	int key_count ;
	GList *keys;	//keys are stored here in the order given in the input file
			///and these can be used in hash table to get the same
			//order while printing the ssf
	GHashTable* hash ;
};
typedef struct fs_struct fs_struct;



struct list_of_nodes {				//list of nodes
	int size;			//list size
	node **l;			//list of node pointers
};
typedef struct list_of_nodes list_of_nodes;
```

Functions :-

```
node * create_tree();
/*returns the root node pointer of the new tree*/

node * create_node (node *parent);
/*returns the node pointer created*/

node * create_node_with_attr(char tkn[], char tag[], char fs[], node *parent);
/*creates a node with attributes mentioned and returns its pointer*/

or_node * create_or_node ();
/*creates a new or_node and returns its pointer*/

fs_struct *create_fs_struct();
/*creates a new fs_struct and returns its pointer*/

void int_to_str(int num,char *str);
/*converts the integer num to a string str*/

void read_ssf_from_file (node *tree, char filename[]);
/*reads the from file into node tree. */

void read_ssf(node *tree, FILE *f);
/*reads the SSF from the file specified by f*/

or_node * read_or_node( char str[]);
/* Builts the or_node form string str and returns it */

fs_struct *read_fs( char str[]);
/* Builts the fs_struct from string str and returns it */

char * make_or_node_to_string ( or_node *OR);
/*convert or_node OR to string format and returns the string */

char *make_fs_struct_to_string ( fs_struct *fs);
/*convert fs_struct fs to string format and returns the string */

char *get_fields (node *N );
/* Gives all the fields in string format and returns the string */

void print_tree(node *tree);
/*Prints the node "tree" in SSF format with index starting from 1*/
```

```
void print_node_without_index(node *N);
/*prints the node "N" without index*/


void print_node(node *tree ,char ind[]);
/*Prints the node "tree" in ssf format with index starting from ind */

void print_attr_of_node(node *N);
/*Prints the attributes of node "N" */

void print_attr_of_or_node(or_node *OR);
/*Prints the attributes of OR node */


int insert_node_into_position(node *parent, node *child, int position);
/*inserts the child at the given position.
*If posistion > (parent->child_count) it will insert be inserted at the end*/


char *get_field(node *N, int n );
/*Gives a nth field of a given node.*/

node *get_nth_child(node *N, int n);
/*returns the nth child of the node N*/

node *get_next_node(node *N);
/*Returns next sibling of a given node.*/

node *get_parent(node *N);
/*Returns parent of a given node.*/

node *get_previous_node(node *N);
/*Returns previous sibling of a given node.*/

int modify_field(node *N, int number,char str[]);
/*Modifies a particular field of a given node.
*returns 1 on success and 0 on failure
*You can also modify or_node here */

int delete_node(node *N);
/*returns 1 on succefull delition else 0 */

list_of_nodes * create_list_of_nodes(int size);
/* creates list of nodes and returns its pointer*/
```

```
list_of_or_nodes * create_list_of_or_nodes(int size);
/* creates list of or_nodes*/

list_of_fs_structs *create_list_of_fs_structs(int size);
/*creates list of fs structures */

int match(const char *string, char *pattern);
/*returns 1 if pattern matches string else 0 */

list_of_nodes *make_list_of_nodes_from_glist(GList *L);
/*Creates list of nodes from list L */

list_of_nodes *getchildren( node *Node );
/*returns a lsit containing childrens of this node */

GList *tmp_getleaves(node *Node, GList *L);
/*gets leaves to List L...Internal function....users can ignore*/

list_of_nodes *getleaves (node *Node);
/*gets all leaves of this node and returns a list_of_nodes containing leaves */

int tmp_count_leaf_nodes(node *tree, int sum);
/*a recursive function used by count)leaf_nodes */

int count_leaf_nodes(node *tree);
/*Counts the number of leaf nodes of the node "tree" */

list_of_nodes *getleaves_child (node *Node );
/*gets the childs which are also the leaves of the node*/

GList *tmp_get_nodes(int num, char str[], node *tree, GList *L) ;
/* internal function to get a list of nodes with particular field value
*Note:implemented using Depth first search*/

list_of_nodes *get_nodes(int fieldnumber, char str[], node *tree);
/*gets nodes which have particular field values in a tree/sub tree
*--Note Feature structure is "NOT INCLUDED" here*/

GList *tmp_get_pattern (int num, char pattern[], node *tree, GList *L);
/*recusive function used by get_pattern. Returns a list of nodes with specified field matching the
pattern*/


list_of_nodes *get_pattern(int fieldnumber, char pattern[], node *tree);
```

```
/*gets nodes of the node "tree" with the specified field matching the pattern
*--Note Feature structure is "INCLUDED" here*/







/*Note: For the below functions
*attribute "af" case is not specially handled.
*To do operations that include the attr "af" the user should deal with indvidual attributes that are
encoded in "af"*/


int add_attr_val(or_node *OR, char attr[], char value[]);
/*Inserts attr values in all the fs's present in or_node
*if any fs is already having attr then it is updated.
*Returns 0 on failure and 1 on success.*/


int add_attr_val_2(fs_struct *fs, char attr[], char value[]);
/*This inserts attr in a particular fs
*If fs already contains attr, then 0 (error) is returned.
*Else attr is inserted & 1 is returned.*/

int update_attr_val(or_node *OR, char attr[], char value[]);
/*This updates the "attr" in all the fs's of the or_node "OR" which contains the key "attr"*/

int update_attr_val_2(fs_struct *fs, char attr[], char value[]);
/*update attr value in the fs.
*If fs does not contain attr 0 is returned.*/

int del_attr(or_node *OR, char attr[]);
/*Deletes attr from all the fs's of or_node*/

int delete_attr_2(fs_struct *fs, char attr[]);
/*Deletes attr from fs_struct.
*if no key with attr is found 0 is returned. */

char *get_attr_val(fs_struct *fs, char attr[]);
/* Get the attr value */



GList * tmp_get_nodes_with_attr_val(node *N, char attr[], char val[], GList *tmp);
```

```
/*a recursive call used by get_nodes_with_attr_val().
Returns a list of nodes satisfying the condition*/


list_of_nodes *get_nodes_with_attr_val(node *N, char attr[], char val[]);
/*get a list of nodes with or_node (fs) attrs equal to val */

void delete_fs_struct_from_or_node(or_node *OR, int n);
/* Deletes nth fs from or_node */

void add_fs_struct_to_or_node (or_node *OR, fs_struct *fs);
/* add the fs_struct fs to or_node */
```