**How to use the Standard Collection Framework in your program:**

Download all the files in the “Code” folder and keep them in the same folder as the program you are going to use them in. Use **#include “std\_cf.h”** at the head of the C program.

Disclaimer: This is not a standard input output program, it is a library which helps in using some common data structures easily and effectively (like stl in C++ and Collection Framework in Java).

A demonstration program of this library in use is [this](https://drive.google.com/file/d/1dr1Vld80AMQj1z9wW_bqwUXBa49NiIXs/view?usp=sharing).

Now, you can use the given data structures like this-

**ArrayList:**

**ArrayList \*arr = NewAL(initial length);**

//declaring and initializing the list

**int append(ArrayList \*arraylist, ArrayListValue data);**

// add at the back of list

**int prepend(ArrayList \*arraylist, ArrayListValue data);**

//add at front of list

**void \_remove(ArrayList \*arraylist, unsigned int index);**

// remove from the list at a particular index

**void \_remove\_range(ArrayList \*arraylist, unsigned int index,**

**unsigned int length);**

//remove a range of values

**int index\_of(ArrayList \*arraylist, ArrayList Equals Func callback, ArrayListValue data);**

//find the index of data

**void \_free(ArrayList \*arraylist);**

//free the memory of arraylist

**void arraylist\_clear(ArrayList \*arraylist);**

//clear the arraylist

**Double-ended Queue:**

//declaration and initialization

**Queue \*q = queue\_new();**

//functions for operations at head

**int queue\_push\_head(Queue \*queue, QueueValue data);**

**QueueValue queue\_pop\_head(Queue \*queue);**

**QueueValue queue\_peek\_head(Queue \*queue);**

//functions for operations at tail

**int queue\_push\_tail(Queue \*queue, QueueValue data);**

**QueueValue queue\_pop\_tail(Queue \*queue);**

**QueueValue queue\_peek\_tail(Queue \*queue);**

//miscellaneous functions

**void queue\_free(Queue \*queue);**

**int queue\_is\_empty(Queue \*queue);**

**Stack:**

//declaration and initialization

**Stack \*st = stack\_new();**

//basic functions

**int stack\_push(Stack \*stack, StackValue data);**

**StackValue stack\_pop(Stack \*stack);**

**StackValue stack\_peek(Stack \*stack);**

//miscellaneous functions

**void stack\_free(Stack \*stack);**

**int stack\_size(Stack \*stack);**

**int stack\_is\_empty(Stack \*stack);**

**Heap:**

**Heap \*hp = heap\_new(Heap\_type,comapre\_func);**

//declaring and initializing the list

**void \_free(Heap \*heap);**

//removing the heap from memory

**int insert(Heap \*heap, datatype value);**

//inserting the new element in heap

**datatype \_remove(Heap \*heap);**

//removing the element from heap

**unsigned int num\_entries(Heap \*heap);**

//total number of values in heap

**Singly Linked List:**

**SListEntry \*l;**

**l = malloc(sizeof(SListEntry));**

**l = NULL;**

//declaring and initializing the list

**SListEntry \*slist\_prepend(SListEntry \*\*list, SListValue data);**

//prepend Elements to the beginning of the list

**SListEntry \*slist\_append(SListEntry \*\*list, SListValue data);**

Append elements to the end of the list

**unsigned int slist\_length(SListEntry \*list);**

//compute length(number of elements) of the list

**SListValue \*slist\_to\_array(SListEntry \*list);**

//convert the list to an array

**SListEntry \*slist\_nth\_entry(SListEntry \*list, unsigned int n);**

//find nth element of the list

**int slist\_remove\_entry(SListEntry \*\*list, SListEntry \*entry);**

//remove particular entry(element) from the list

**Doubly Linked List:**

**ListEntry \*l;**

**l = malloc(sizeof(ListEntry));**

**l = NULL;**

//declaring and initializing the list

**ListEntry \*list\_prepend(ListEntry \*\*list, ListValue data);**

//prepend Elements to the beginning of the list

**ListEntry \*list\_append(ListEntry \*\*list, ListValue data);**

Append elements to the end of the list

**unsigned int list\_length(ListEntry \*list);**

//compute length(number of elements) of the list

**ListValue \*list\_to\_array(ListEntry \*list);**

//convert the list to an array

**ListEntry \*list\_nth\_entry(ListEntry \*list, unsigned int n);**

//find nth element of the list

**int list\_remove\_entry(ListEntry \*\*list, ListEntry \*entry);**

//remove particular entry(element) from the list