**TITLE OF LAB: (IMPLEMENTATION OF LISTS USING ARRAYS)**

**LAB REPORT NO.06**



**Spring 2022**

**CSE-210L Data Structures and Algorithm Lab**

Submitted by

Name:: **Safi Ullah Khan**

Registration No. **20PWCSE1943**

Class Section: **B**

“On my honor, as student of University of Engineering and Technology, I have neither given nor received unauthorized assistance on this academic work.”

Student Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Submitted to:

**Dr. Khurram Shehzad Khattak**

(Friday, July 29th, 2022)

**Department of Computer Systems Engineering**

**University of Engineering and Technology, Peshawar**

**OBJECTIVES OF THE LAB**

‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐

In this lab, we will learn about the Data Structure, lists and their useful operations.

‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐

## **Task 01**

Implement LIST using arrays. It should perform following operations

* Create (Creates LIST)
* Add (Adds item to the LIST at given index)
* Remove (Removes item from the List at provided index)
* waSize (Determines Size of the LIST)
* IsEmpty (Determines if LIST is empty or not)
* Get (Retrieves an Item from the LIST)
* End (Returns end of the LIST)
* Start (Return start of the LIST)

## **Task 02**

Debug following code.

/\* array implementation of LIST ADT \*/

#include <stdio.h>

#include <math.h>

#include <string.h>

#define MAX\_LIST\_SIZE 100

#define FALSE 0

#define TRUE 1

typedef struct {

int number;

char \*string;

} ELEMENT\_TYPE;

typedef struct {

int last;

ELEMENT\_TYPE a[MAX\_LIST\_SIZE];

} LIST\_TYPE;

typedef int WINDOW\_TYPE;

/\*\* position following last element in a list \*\*\*/

WINDOW\_TYPE end(LIST\_TYPE \*list) {

return(list->last+1);

}

/\*\*\* empty a list \*\*\*/

WINDOW\_TYPE empty(LIST\_TYPE \*list) {

list->last = -1;

return(end(list));

}

/\*\*\* test to see if a list is empty \*\*\*/

int is\_empty(LIST\_TYPE \*list) {

if (list->last == -1)

return(TRUE);

else

return(FALSE)

/\*\*\* position at first element in a list \*\*\*/

WINDOW\_TYPE first(LIST\_TYPE \*list) {

if (is\_empty(list) == FALSE) {

return(0);

else

return(end(list));

}

/\*\*\* position at next element in a list \*\*\*/

WINDOW\_TYPE next(WINDOW\_TYPE w, LIST\_TYPE \*list) {

if (w == last(list)) {

return(end(list));

else if (w == end(list)) {

error(“can’t find next after end of list”); }

else {

return(w+1);

} }

/\*\*\* position at previous element in a list \*\*\*/

WINDOW\_TYPE previous(WINDOW\_TYPE w, LIST\_TYPE \*list) {

if (w != first(list)) {

return(w-1);

else {

error(“can’t find previous before first element of

list”);

return(w); } }

/\*\*\* position at last element in a list \*\*\*/

WINDOW\_TYPE last(LIST\_TYPE \*list) {

return(list->last);}