**IMPLEMENTATION OF LIST USING ARRAYS**

**LAB # 06**



**Data Structures & Algorithms**

Submitted by: **Shah Raza**

Registration No: **18PWCSE1658**

Class Section: **B**

“On my honor, as a 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**

**Department of Computer Systems Engineering**

**University of Engineering and Technology, Peshawar**

**Lab Objectives:**

To learn about the data structure, lists and their useful operations.

**Task # 1:**

Implement LIST using arrays.

**Code:**

#include <iostream>

using namespace std;

typedef struct {

int RegNo=0;

string Name="Default";

int Marks=0;

} Node;

Node\* CreateList (int S,int \*Last)

{ Node \*Arr= new Node[S];

\*Last=0;

return Arr;

}

Node\* Add(int \*Last,int index,int S,Node \*Arr,Node nd)

{ if(\*Last<0)

cout<<"List does not exist";

else if(\*Last >= S)

cout<<"List is Full";

else if(index>\*Last)

cout<<"invalid Position";

else

{

for(int i=\*Last;i>=index;i--)

Arr[i+1]=Arr[i];

Arr[index]=nd;

\*Last=\*Last+1;

}

return Arr;

}

Node\* Remove(int \*Last,int index,int S,Node \*Arr)

{

if(\*Last<0)

cout<<"List does not exist";

else if(\*Last == 0)

cout<<"List is Already Empty";

else

{

for(int i=index+1;i<=\*Last;i++)

Arr[i-1]=Arr[i];

\*Last=\*Last-1;

}

return Arr;

}

void Size(int \*Last)

{

cout<<"Size of the List is: "<<\*Last;

}

void Print(int \*Last,Node \*Arr)

{

for(int i=0;i<\*Last;i++)

{

cout<<"\nName: "<<Arr[i].Name;

cout<<"\nRegNo: "<<Arr[i].RegNo<<"\nMarks: "<<Arr[i].Marks<<endl;

}

}

void Get(int index,Node \*Arr)

{

cout<<"Element at index["<<index<<"] is: "<<endl;

cout<<"\nName: ";

cout<<Arr[index].Name<<"\nRegNo: "<<Arr[index].RegNo;

cout<<"\nMarks: "<<Arr[index].Marks<<endl;

}

void IsEmpty(int \*Last)

{

if(\*Last<=0)

cout<<"List is Empty"<<endl;

else

cout<<"List is not Empty"<<endl;

}

int main()

{

Node \*Ar;

Node nd;

int choice;

int S,index,Last=-1;

do{

cout<<"\n---------------------"<<endl;

cout<<"\n1) Create List \n2) Add Node \n3) Remove Node";

cout<<"\n4) Print\n5) Get Element\n6) Size of List";

cout<<"\n7) Is list Empty?\n\n0)Quit"<<endl;

cin>>choice;

cout<<"\n";

switch (choice){

case 1:

cout<<"Enter the size of the list: ";

cin>>S;

Ar=CreateList(S,&Last);

break;

case 2:

cout<<"Enter the Name, RegNo and Marks: "<<endl;

cin>>nd.Name>>nd.RegNo>>nd.Marks;

cout<<"Enter the Index Number: ";

cin>>index;

Ar=Add(&Last,index,S,Ar,nd);

break;

case 3:

cout<<"Enter the Index Number: ";

cin>>index;

Ar=Remove(&Last,index,S,Ar);

break;

case 4:

Print(&Last,Ar);

break;

case 5:

cout<<"Enter the Index Number: ";

cin>>index;

Get(index,Ar);

break;

case 6:

Size(&Last);

break;

case 7:

IsEmpty(&Last);

break;

}

}while (choice != 0);

return 0;

}

**Pseudo Code:**

* Ask the user to choose an option from the menu
* Add Entry Function

{

if last is less than 0

then print "List does not exist"

else if last is greater than or equal to size

then print "List is Full"

else if index is greater than last

then print "invalid Position"

else

for(i equal to last; i is greater than or equal to index; i--)

{ Swap Array[i+1] and Array[i] }

Arr[index] equal to entered element;

Increment last

}

return Array

}

* Remove Entry Function

{

if last is less than 0

then print "List does not exist"

else if last is equal to 0

then print "List is Already Empty"

else

for(i equal to index+1 ; i is less than or equal to last ; i++)

{ Swap Array[i-1] and Array[i] }

Decrement last

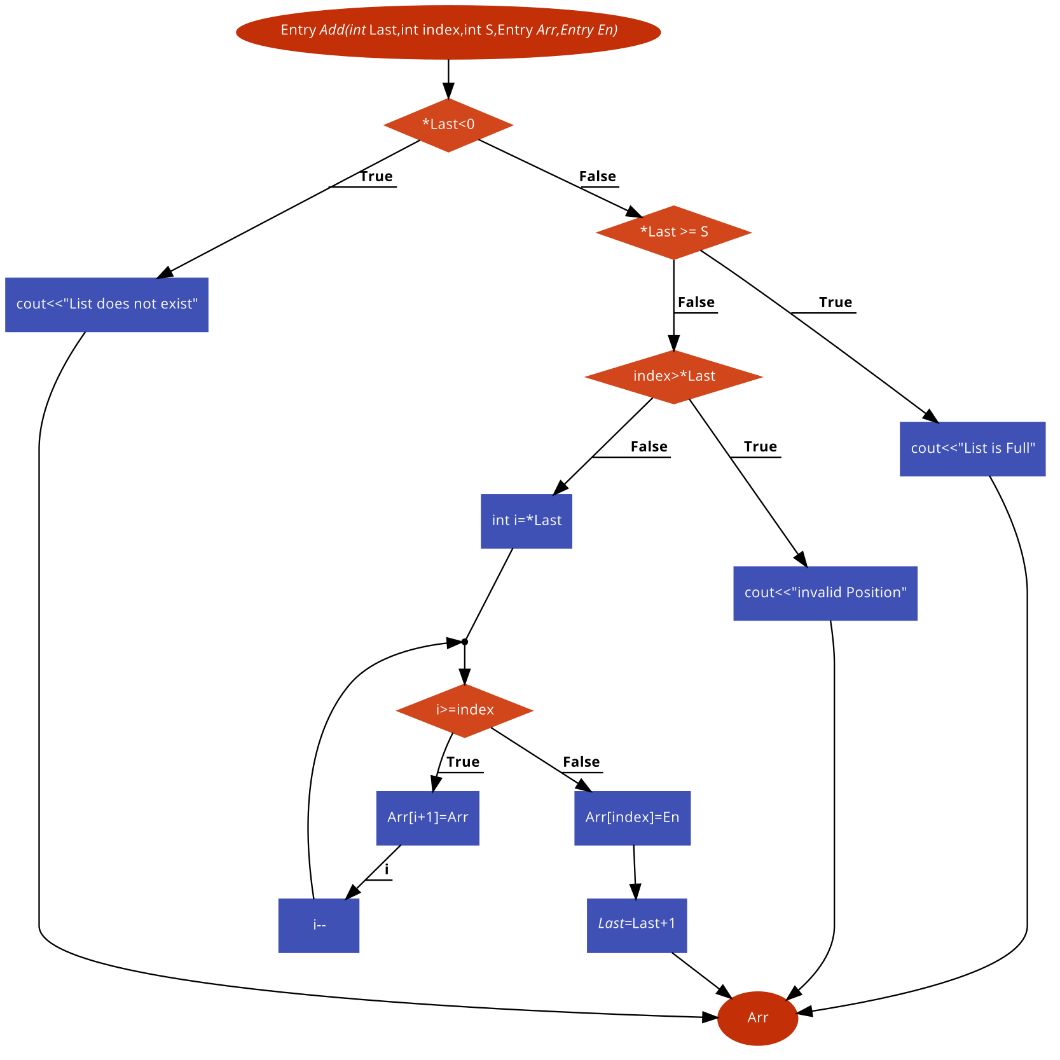
}

return Array

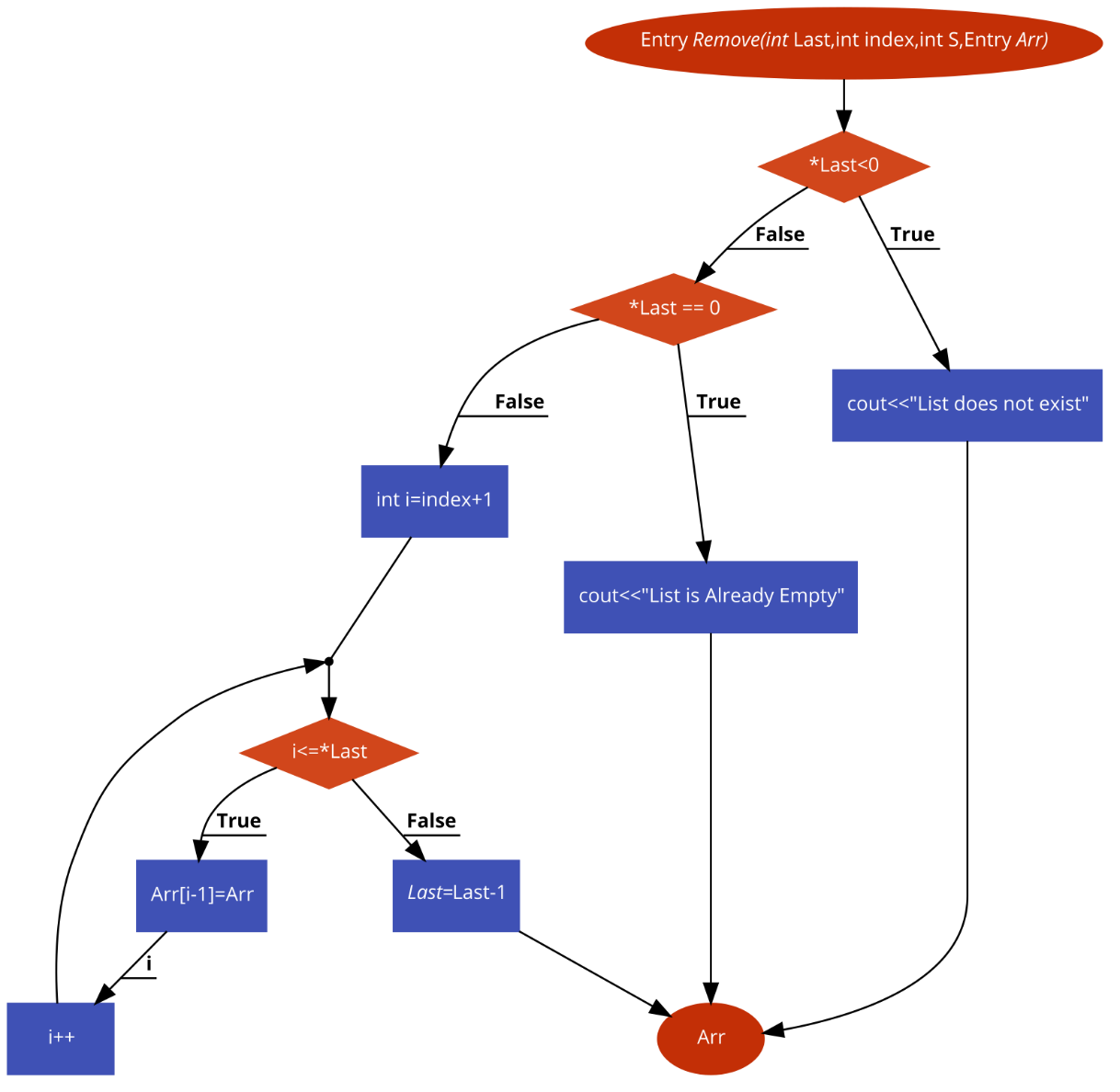
}

**Flow Chart:**

Add Entry:

****

Remove Entry:

****

**Task # 2:**

Debug the given code.

**Debugged 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);

}

int error(char \*s) { //Missing Function Added

printf("Error: %s\n", s);

return 0;

}

/\*\*\* 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) ; } // Missing semi colon and Bracket added

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

WINDOW\_TYPE last(LIST\_TYPE \*list) { //Function moved up so that other

return(list->last);} //functions can call it

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

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

if (w == last(list)) {

return(end(list)); } // Missing Bracket added

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); } // Missing Bracket added

else {

error(“can’t find previous before first element of list”);

return(w); } }