

Experiment #1		Student ID	
Date		Student Name	@KLWKS_BOT THANOS

Experiment Title: To implement basic Programs on Arrays and Linked Lists.

Aim/Objective: To understand the concept and implementation of programs on arrays and linked lists.

Description: The students will be able to implement programs on Arrays and Linked Lists.

Pre-Requisites:

Knowledge: Arrays and Linked Lists.

Tools: Code Blocks/Eclipse IDE

Pre-Lab:

1. An *array* is a type of data structure that stores elements of the same type in a contiguous block of memory. In an array A, of size N, each memory location has some unique index i , (where $0 \leq i < N$), that can be referenced as $A[i]$ or A_i .

Reverse an array of integers.

Example

$A = [1, 2, 3]$

Return $[3, 2, 1]$

Function Description

Complete the function *reverseArray* in the editor below.

reverseArray has the following parameter(s):

- *int A[n]*: the array to reverse

Returns

- *int[n]*: the reversed array

Input Format

The first line contains an integer N, the number of integers in A.

The second line contains N space-separated integers that make up A.

Constraints

$$1 \leq N \leq 10^3$$

$$1 \leq A[i] \leq 10^4, \text{ where } A[i] \text{ is the } i^{\text{th}} \text{ integer in A.}$$

Course Title	Advanced Algorithms & Data Structures	Academic Year: 2024-25
Course Code	23CS03HF	1 Page

Experiment #1		Student ID	
Date		Student Name	@KLWKS_BOT THANOS

Sample Input 1

4
1 4 3 2

Sample Output 1

2 3 4 1

• Procedure/Program:

```
#include <stdio.h>
```

```
int main() {
```

```
    int N;
```

```
    printf("Enter the size of the array: ");
```

```
    scanf("%d", &N);
```

```
    int A[N];
```

```
    printf("Enter the elements of the array:\n");
```

```
    for (int i = 0; i < N; i++) {
```

```
        scanf("%d", &A[i]);
```

```
    }
```

```
    printf("Reversed Array: ");
```

```
    for (int i = N - 1; i >= 0; i--) {
```

Course Title	Advanced Algorithms & Data Structures	Academic Year: 2024-25
Course Code	23CS03HF	2 Page

Experiment #1		Student ID	
Date		Student Name	@KLWKS_BOT THANOS

```

    printf("%d ", A[i]);

}

return 0;

}

```

- **Data and Results:**

Data

The array contains integers with a specified size and order.

Result

Reversed array displays integers in the opposite order of input.

Course Title	Advanced Algorithms & Data Structures	Academic Year: 2024-25
Course Code	23CS03HF	3 Page

Experiment #1		Student ID	
Date		Student Name	@KLWKS_BOT THANOS

- **Analysis and Inferences:**

Analysis

Reversing ensures the last element becomes the first, iteratively.

Inferences

The reverse operation is efficient and retains original data integrity.

2. There is a collection of input strings and a collection of query strings. For each query string, determine how many times it occurs in the list of input strings. Return an array of the results.

Example

```
stringList = ['ab', 'ab', 'abc']
```

```
queries = ['ab', 'abc', 'bc']
```

There are 2 instances of 'ab', 1 of 'abc' and 0 of 'bc'. For each query, add an element to the return array, *results* = [2,1,0].

Function Description

Complete the function `matchingStrings` in the editor below. The function must return an array of integers representing the frequency of occurrence of each query string in strings.

`matchingStrings` has the following parameters:

string `stringList[n]` – an array of strings to search

string `queries[q]` – an array of query strings

Returns

int[q]: an array of results for each query

Input Format

The first line contains an integer `n`, the size of `stringList[]`.

Each of the next `n` lines contains a string `stringList[i]`.

The next line contains `q`, the size of `queries[]`.

Course Title	Advanced Algorithms & Data Structures	Academic Year: 2024-25
Course Code	23CS03HF	4 Page

Experiment #1		Student ID	
Date		Student Name	@KLWKS_BOT THANOS

Each of the next q lines contains a string queries[i] .

Constraints

$$1 \leq n \leq 10^3$$

$$1 \leq q \leq 10^3,$$

$$1 \leq \text{stringList}[i], \text{queries}[i] \leq 20$$

• Procedure/Program:

```
#include <stdio.h>
#include <string.h>
```

```
int main() {
    int n, q;

    printf("Enter the size of stringList: ");
    scanf("%d", &n);

    char stringList[n][100];
    printf("Enter the strings for stringList:\n");
    for (int i = 0; i < n; i++) {
        scanf("%s", stringList[i]);
    }
```

```
printf("Enter the size of queries: ");
scanf("%d", &q);
```

```
char queries[q][100];
printf("Enter the strings for queries:\n");
for (int i = 0; i < q; i++) {
```

Course Title	Advanced Algorithms & Data Structures	Academic Year: 2024-25
Course Code	23CS03HF	5 Page

Experiment #1		Student ID	
Date		Student Name	@KLWKS_BOT THANOS

```

    scanf("%s", queries[i]);
}

int results[q];
for (int i = 0; i < q; i++) {
    results[i] = 0;
}

for (int i = 0; i < q; i++) {
    for (int j = 0; j < n; j++) {
        if (strcmp(queries[i], stringList[j]) == 0) {
            results[i]++;
        }
    }
}

printf("Results: ");
for (int i = 0; i < q; i++) {
    printf("%d ", results[i]);
}

return 0;
}

```

Course Title	Advanced Algorithms & Data Structures	Academic Year: 2024-25
Course Code	23CS03HF	6 Page

Experiment #1		Student ID	
Date		Student Name	@KLWKS_BOT THANOS

- **Data and Results:**

Data

The input consists of strings and queries for frequency comparison.

Result

Counts of each query's occurrences in the string list returned.

- **Analysis and Inferences:**

Analysis

Query strings are checked against input list using nested iteration.

Inferences

Higher frequency queries indicate repeated patterns in the string list.

Course Title	Advanced Algorithms & Data Structures	Academic Year: 2024-25
Course Code	23CS03HF	7 Page

Experiment #1		Student ID	
Date		Student Name	@KLWKS_BOT THANOS

In-Lab:

1. Given pointers to the heads of two sorted linked lists, merge them into a single, sorted linked list. Either head pointer may be null meaning that the corresponding list is empty.

Example

headA refers to 1 -> 3 -> 7 -> NULL

headB refers to 1 -> 2 -> NULL

The new list is 1 -> 2 -> 3 -> 7 -> NULL

Function Description:

Complete the mergeLists function in the editor below.

mergeLists has the following parameters:

- SinglyLinkedListNode pointer headA: a reference to the head of a list
- SinglyLinkedListNode pointer headB: a reference to the head of a list

Returns

SinglyLinkedListNode pointer: a reference to the head of the merged list

Input Format

The first line contains an integer t , the number of test cases.

The format for each test case is as follows:

The first line contains an integer n , the length of the first linked list.

The next n lines contain an integer each, the elements of the linked list.

The next line contains an integer m , the length of the second linked list.

The next n lines contain an integer each, the elements of the second linked list.

Sample Input:

```
1
3
1
2
3
2
3
```

Course Title	Advanced Algorithms & Data Structures	Academic Year: 2024-25
Course Code	23CS03HF	8 Page

Experiment #1		Student ID	
Date		Student Name	@KLWKS_BOT THANOS

4

Sample Output:

1 2 3 3 4

Explanation:

The first linked list is: 1 -> 2 -> 3 -> NULL

The second linked list is: 3 -> 4 -> NULL

Hence, the merged linked list is: 1 -> 2 -> 3 -> 3 -> 4 -> NULL

• Procedure/Program:

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
struct SinglyLinkedListNode {
    int data;
    struct SinglyLinkedListNode* next;
};
```

```
void printList(struct SinglyLinkedListNode* head) {
    while (head != NULL) {
        printf("%d ", head->data);
        head = head->next;
    }
    printf("\n");
}
```

```
int main() {
    int t;
    scanf("%d", &t);

    while (t > 0) {
        t--;

        int n, m, data;
        struct SinglyLinkedListNode *headA = NULL, *headB = NULL, *currentA, *currentB,
        *dummy, *tail;
```

Course Title	Advanced Algorithms & Data Structures	Academic Year: 2024-25
Course Code	23CS03HF	9 Page

Experiment #1		Student ID	
Date		Student Name	@KLWKS_BOT THANOS

```

scanf("%d", &n);
if (n > 0) {
    scanf("%d", &data);
    headA = (struct SinglyLinkedListNode*)malloc(sizeof(struct SinglyLinkedListNode));
    headA->data = data;
    headA->next = NULL;
    currentA = headA;
    for (int i = 1; i < n; i++) {
        scanf("%d", &data);
        currentA->next = (struct SinglyLinkedListNode*)malloc(sizeof(struct
SinglyLinkedListNode));
        currentA = currentA->next;
        currentA->data = data;
        currentA->next = NULL;
    }
}

scanf("%d", &m);
if (m > 0) {
    scanf("%d", &data);
    headB = (struct SinglyLinkedListNode*)malloc(sizeof(struct SinglyLinkedListNode));
    headB->data = data;
    headB->next = NULL;
    currentB = headB;
    for (int i = 1; i < m; i++) {
        scanf("%d", &data);
        currentB->next = (struct SinglyLinkedListNode*)malloc(sizeof(struct
SinglyLinkedListNode));
        currentB = currentB->next;
        currentB->data = data;
        currentB->next = NULL;
    }
}

dummy = (struct SinglyLinkedListNode*)malloc(sizeof(struct SinglyLinkedListNode));
tail = dummy;
while (headA != NULL && headB != NULL) {

```

Course Title	Advanced Algorithms & Data Structures	Academic Year: 2024-25
Course Code	23CS03HF	10 Page

Experiment #1		Student ID	
Date		Student Name	@KLWKS_BOT THANOS

```

    if (headA->data <= headB->data) {
        tail->next = headA;
        headA = headA->next;
    } else {
        tail->next = headB;
        headB = headB->next;
    }
    tail = tail->next;
}
tail->next = (headA != NULL) ? headA : headB;
struct SinglyLinkedListNode* mergedHead = dummy->next;

printList(mergedHead);
}

return 0;
}

```

Course Title	Advanced Algorithms & Data Structures	Academic Year: 2024-25
Course Code	23CS03HF	11 Page

Experiment #1		Student ID	
Date		Student Name	@KLWKS_BOT THANOS

- **Data and Results:**

Data

Two sorted linked lists are given as input for merging.

Result

A single sorted linked list is returned as the output.

- **Analysis and Inferences:**

Analysis

The merging process compares nodes and builds the sorted list.

Inferences

Efficient merging preserves order and handles empty lists seamlessly.

Course Title	Advanced Algorithms & Data Structures	Academic Year: 2024-25
Course Code	23CS03HF	12 Page

Experiment #1		Student ID	
Date		Student Name	@KLWKS_BOT THANOS

2. You are given the pointer to the head node of a sorted linked list, where the data in the nodes is in ascending order. Delete nodes and return a sorted list with each distinct value in the original list.

The given head pointer may be null indicating that the list is empty.

Example

head refers to the first node in the list.

1->2->2->3->3->3->3->NULL

Remove 1 of the 2 data values and return head pointing to the revised list, 1->2->3->NULL

Function Description

Complete the removeDuplicates function in the editor below.

removeDuplicates has the following parameter:

- SinglyLinkedListNode pointer head: a reference to the head of the list

Returns

- SinglyLinkedListNode pointer: a reference to the head of the revised list

Input Format

The first line contains an integer t , the number of test cases.

The format for each test case is as follows:

The first line contains an integer n , the number of elements in the linked list.

Each of the next n lines contains an integer, the data value for each of the elements of the linked list.

Constraints

$$1 \leq t \leq 10$$

$$1 \leq n \leq 1000$$

$$1 \leq \text{list}[i] \leq 1000$$

Sample Input

STDIN Function

1 t = 1

5 n = 5

1 data values = 1, 2, 2, 3, 4

2

2

3

4

Sample Output

1 2 3 4

Course Title	Advanced Algorithms & Data Structures	Academic Year: 2024-25
Course Code	23CS03HF	13 Page

Experiment #1		Student ID	
Date		Student Name	@KLWKS_BOT THANOS

• **Procedure/Program:**

```
#include <stdio.h>
#include <stdlib.h>

struct SinglyLinkedListNode {
    int data;
    struct SinglyLinkedListNode* next;
};

void printList(struct SinglyLinkedListNode* head) {
    while (head != NULL) {
        printf("%d ", head->data);
        head = head->next;
    }
    printf("\n");
}

int main() {
    int t;
    scanf("%d", &t);

    while (t > 0) {
        t--;

        int n;
        struct SinglyLinkedListNode *head = NULL, *current = NULL;

        scanf("%d", &n);
        if (n > 0) {
            int data;
            scanf("%d", &data);
            head = (struct SinglyLinkedListNode*)malloc(sizeof(struct SinglyLinkedListNode));
            head->data = data;
            head->next = NULL;
            current = head;
        }
    }
}
```

Course Title	Advanced Algorithms & Data Structures	Academic Year: 2024-25
Course Code	23CS03HF	14 Page

Experiment #1		Student ID	
Date		Student Name	@KLWKS_BOT THANOS

```

    for (int i = 1; i < n; i++) {
        scanf("%d", &data);
        current->next = (struct SinglyLinkedListNode*)malloc(sizeof(struct
SinglyLinkedListNode));
        current = current->next;
        current->data = data;
        current->next = NULL;
    }
}

current = head;
while (current != NULL && current->next != NULL) {
    if (current->data == current->next->data) {
        current->next = current->next->next;
    } else {
        current = current->next;
    }
}

printList(head);
}

return 0;
}

```

Course Title	Advanced Algorithms & Data Structures	Academic Year: 2024-25
Course Code	23CS03HF	15 Page

Experiment #1		Student ID	
Date		Student Name	@KLWKS_BOT THANOS

- **Data and Results:**

Data

Sorted linked list is provided with potential duplicate values.

Result

The revised list contains only distinct values in sorted order.

- **Analysis and Inferences:**

Analysis

Duplicates are removed by comparing adjacent nodes in the list.

Inferences

Efficient solution relies on sorted order, removing duplicates in-place.

Course Title	Advanced Algorithms & Data Structures	Academic Year: 2024-25
Course Code	23CS03HF	16 Page

Experiment #1		Student ID	
Date		Student Name	@KLWKS_BOT THANOS

Post-Lab:

1. How will you modify a linked list of integers so that all even numbers appear before all odd numbers in the modified linked list? Also, keep the even and odd numbers in the same order.

Example:

Input: 17->15->8->12->10->5->4->1->7->6->NULL

Output: 8->12->10->4->6->17->15->5->1->7->NULL.

- **Procedure:**

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
struct SinglyLinkedListNode {
```

```
    int data;
```

```
    struct SinglyLinkedListNode* next;
```

```
};
```

```
void printList(struct SinglyLinkedListNode* head) {
```

```
    while (head != NULL) {
```

```
        printf("%d ", head->data);
```

```
        head = head->next;
```

```
    }
```

```
    printf("\n");
```

```
}
```

```
int main() {
```

```
    int n;
```

Course Title	Advanced Algorithms & Data Structures	Academic Year: 2024-25
Course Code	23CS03HF	17 Page

Experiment #1		Student ID	
Date		Student Name	@KLWKS_BOT THANOS

```
printf("Enter the number of elements: ");
```

```
scanf("%d", &n);
```

```
struct SinglyLinkedListNode* head = NULL;
```

```
struct SinglyLinkedListNode* current = NULL;
```

```
printf("Enter the elements: ");
```

```
for (int i = 0; i < n; i++) {
```

```
    int data;
```

```
    scanf("%d", &data);
```

```
    struct SinglyLinkedListNode* newNode = (struct  
SinglyLinkedListNode*)malloc(sizeof(struct SinglyLinkedListNode));
```

```
    newNode->data = data;
```

```
    newNode->next = NULL;
```

```
    if (head == NULL) {
```

```
        head = newNode;
```

```
        current = head;
```

```
    } else {
```

```
        current->next = newNode;
```

```
        current = current->next;
```

```
    }
```

```
}
```

```
struct SinglyLinkedListNode *evenHead = NULL, *evenTail = NULL;
```

```
struct SinglyLinkedListNode *oddHead = NULL, *oddTail = NULL;
```

Course Title	Advanced Algorithms & Data Structures	Academic Year: 2024-25
Course Code	23CS03HF	18 Page

Experiment #1		Student ID	
Date		Student Name	@KLWKS_BOT THANOS

```

current = head;
while (current != NULL) {
    if (current->data % 2 == 0) {
        if (evenHead == NULL) {
            evenHead = evenTail = current;
        } else {
            evenTail->next = current;
            evenTail = evenTail->next;
        }
    } else {
        if (oddHead == NULL) {
            oddHead = oddTail = current;
        } else {
            oddTail->next = current;
            oddTail = oddTail->next;
        }
    }
    current = current->next;
}

```

```

if (evenTail != NULL) {
    evenTail->next = oddHead;
}

if (oddTail != NULL) {
    oddTail->next = NULL;
}

```

Course Title	Advanced Algorithms & Data Structures	Academic Year: 2024-25
Course Code	23CS03HF	19 Page

Experiment #1		Student ID	
Date		Student Name	@KLWKS_BOT THANOS

```
head = (evenHead != NULL) ? evenHead : oddHead;
```

```
printList(head);
```

```
return 0;
```

```
}
```

- **Data and Results:**

Data:

The input is a linked list with unsorted integers.

Result:

The output is a reordered linked list of integers.

- **Analysis and Inferences:**

Analysis:

Even numbers are placed before odd numbers, preserving relative order.

Inferences:

Reordering preserves structure; even elements come before odd elements.

Course Title	Advanced Algorithms & Data Structures	Academic Year: 2024-25
Course Code	23CS03HF	20 Page

Experiment #1		Student ID	
Date		Student Name	@KLWKS_BOT THANOS

2. Given a linked list that is sorted based on absolute values. Sort the list based on actual values.

Examples:

Input-1 : 1 -> -10

Output-1: -10 -> 1

Input-2 : 1 -> -2 -> -3 -> 4 -> -5

Output-2: -5 -> -3 -> -2 -> 1 -> 4

- **Procedure:**

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
struct SinglyLinkedListNode {
```

```
    int data;
```

```
    struct SinglyLinkedListNode* next;
```

```
};
```

```
void printList(struct SinglyLinkedListNode* head) {
```

```
    while (head != NULL) {
```

```
        printf("%d ", head->data);
```

```
        head = head->next;
```

```
    }
```

```
    printf("\n");
```

```
}
```

```
int main() {
```

```
    int n;
```

Course Title	Advanced Algorithms & Data Structures	Academic Year: 2024-25
Course Code	23CS03HF	21 Page

Experiment #1		Student ID	
Date		Student Name	@KLWKS_BOT THANOS

```
scanf("%d", &n);
```

```
struct SinglyLinkedListNode* head = NULL;
```

```
struct SinglyLinkedListNode* current = NULL;
```

```
for (int i = 0; i < n; i++) {
```

```
    int data;
```

```
    scanf("%d", &data);
```

```
    struct SinglyLinkedListNode* newNode = (struct  
SinglyLinkedListNode*)malloc(sizeof(struct SinglyLinkedListNode));
```

```
    newNode->data = data;
```

```
    newNode->next = NULL;
```

```
    if (head == NULL) {
```

```
        head = newNode;
```

```
        current = head;
```

```
    } else {
```

```
        current->next = newNode;
```

```
        current = current->next;
```

```
    }
```

```
}
```

```
struct SinglyLinkedListNode* dummy = (struct SinglyLinkedListNode*)malloc(sizeof(struct  
SinglyLinkedListNode));
```

```
dummy->data = 0;
```

```
dummy->next = head;
```

```
struct SinglyLinkedListNode* prev = dummy;
```

Course Title	Advanced Algorithms & Data Structures	Academic Year: 2024-25
Course Code	23CS03HF	22 Page

Experiment #1		Student ID	
Date		Student Name	@KLWKS_BOT THANOS

```
current = head;
```

```
while (current && current->next) {
    if (current->data > current->next->data) {
        struct SinglyLinkedListNode* temp = current->next;
        current->next = temp->next;
        prev->next = temp;
        temp->next = current;
        prev = dummy;
    } else {
        prev = current;
        current = current->next;
    }
}
```

```
current = dummy->next;
printList(current);
```

```
free(dummy);
```

```
return 0;
```

```
}
```

Course Title	Advanced Algorithms & Data Structures	Academic Year: 2024-25
Course Code	23CS03HF	23 Page

Experiment #1		Student ID	
Date		Student Name	@KLWKS_BOT THANOS

- **Sample VIVA-VOCE Questions (In-Lab):**

1. What is the difference between an array and a linked list?

- **Array:** Fixed size, contiguous memory, direct access via index.
- **Linked List:** Dynamic size, nodes with pointers, sequential access.

2. What are the advantages and disadvantages of using linked lists?

- **Advantages:** Dynamic size, efficient insertions/deletions.
- **Disadvantages:** Extra memory for pointers, slower access time, complex implementation.

3. How will you find the middle element of a singly linked list without iterating the list more than once?

- **Use two pointers: slow (moves 1 step) and fast (moves 2 steps). When fast reaches end, slow is at the middle.**

4. What is the time complexity for accessing an element in an array?

- **$O(1)$ (constant time).**

Course Title	Advanced Algorithms & Data Structures	Academic Year: 2024-25
Course Code	23CS03HF	24 Page

Experiment #1		Student ID	
Date		Student Name	@KLWKS_BOT THANOS

5. How would you reverse an array in-place in linear time and constant space?

- Swap elements from both ends of the array towards the center in $O(n)$ time and $O(1)$ space.

Evaluator Remark (if Any):	Marks Secured ____ out of 50
	Signature of the Evaluator with Date

Evaluator MUST ask Viva-voce prior to signing and posting marks for each experiment.

Course Title	Advanced Algorithms & Data Structures	Academic Year: 2024-25
Course Code	23CS03HF	25 Page