Rajalakshmi Engineering College

Name: Aashira Fathima
Email: 241501004@rajalakshmi.edu.in
Roll no: 241501004
Phone: 9600884785
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 2_CY

Attempt : 1
Total Mark : 30
Marks Obtained : 30

Section 1 : Coding

1. Problem Statement

Ashiq is developing a ticketing system for a small amusement park. The park issues tickets to visitors in the order they arrive. However, due to a system change, the oldest ticket (first inserted) must be revoked instead of the last one.

To manage this, Ashiq decided to use a doubly linked list-based stack, where:

Pushing adds a new ticket to the top of the stack.Removing the first inserted ticket (removing from the bottom of the stack).Printing the remaining tickets from bottom to top.

Input Format

The first line consists of an integer n, representing the number of tickets issued.

The second line consists of n space-separated integers, each representing a ticket number in the order they were issued.

Output Format

The output prints space-separated integers, representing the remaining ticket numbers in the order from bottom to top.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 7
24 96 41 85 97 91 13
Output: 96 41 85 97 91 13

Answer

```
// You are using GCC
#include <stdio.h>

#include <stdlib.h>

struct Node {

    int ticketNumber;

    struct Node* next;

    struct Node* prev;

};


struct Node* createNode(int ticketNumber) {

    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));

    newNode->ticketNumber = ticketNumber;

    newNode->next = NULL;
```

```
    newNode->prev = NULL;
    return newNode;
}
void push(struct Node** head, int ticketNumber) {
    struct Node* newNode = createNode(ticketNumber);
    if (*head == NULL) {
        *head = newNode;
    } else {
        newNode->next = *head;
        (*head)->prev = newNode;
        *head = newNode;
    }
}
void revokeOldestTicket(struct Node** head) {
    if (*head == NULL) {
        return;
    }

    struct Node* temp = *head;
    while (temp->next != NULL) {
```

```
        temp = temp->next;
    }

    if (temp->prev == NULL) {
        free(temp);
        *head = NULL;
    } else {
        temp->prev->next = NULL;
        free(temp);
    }
}

void printTickets(struct Node* head) {
    struct Node* temp = head;

    while (temp != NULL && temp->next != NULL) {
        temp = temp->next;
    }

    while (temp != NULL) {
        printf("%d ", temp->ticketNumber);
        temp = temp->prev;
    }
```

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

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

    struct Node* head = NULL;
    for (int i = 0; i < n; i++) {
        int ticketNumber;
        scanf("%d", &ticketNumber);
        push(&head, ticketNumber);
    }

    revokeOldestTicket(&head);
    printTickets(head);
    struct Node* current = head;
    while (current != NULL) {
        struct Node* next = current->next;
        free(current);
        current = next;
    }
```

```
    return 0;
}
```

Status : Correct **Marks : 10/10**

2. Problem Statement

Sam is learning about two-way linked lists. He came across a problem where he had to populate a two-way linked list and print the original as well as the reverse order of the list. Assist him with a suitable program.

Input Format

The first line of input consists of an integer n, representing the number of elements in the list.

The second line consists of n space-separated integers, representing the elements.

Output Format

The first line displays the message: "List in original order:"

The second line displays the elements of the doubly linked list in the original order.

The third line displays the message: "List in reverse order:"

The fourth line displays the elements of the doubly linked list in reverse order.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 5
1 2 3 4 5
Output: List in original order:
1 2 3 4 5
List in reverse order:

5 4 3 2 1

Answer

```
// You are using GCC
#include <stdio.h>

#include <stdlib.h>


struct Node {

    int data;

    struct Node* next;

    struct Node* prev;

};


struct Node* createNode(int data) {

    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));

    newNode->data = data;

    newNode->next = NULL;

    newNode->prev = NULL;

    return newNode;

}


void append(struct Node** head, int data) {

    struct Node* newNode = createNode(data);

    if (*head == NULL) {

        *head = newNode;
```

```
    } else {
        struct Node* temp = *head;
        while (temp->next != NULL) {
            temp = temp->next;
        }
        temp->next = newNode;
        newNode->prev = temp;
    }
}

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

void printReverseOrder(struct Node* head) {
    struct Node* temp = head;
```

```
    while (temp != NULL && temp->next != NULL) {
        temp = temp->next;
    }

    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->prev;
    }
    printf("\n");
}

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

    struct Node* head = NULL;

    for (int i = 0; i < n; i++) {
        int value;
        scanf("%d", &value);
        append(&head, value);
    }
```

```
    printf("List in original order:\n");
    printOriginalOrder(head);

    printf("List in reverse order:\n");
    printReverseOrder(head);
    struct Node* current = head;
    while (current != NULL) {
        struct Node* next = current->next;
        free(current);
        current = next;
    }

    return 0;
}
```

Status : Correct **Marks : 10/10**

3. Problem Statement

Aarav is working on a program to analyze his test scores, which are stored in a doubly linked list. He needs a solution to input scores into the list and determine the highest score.

Help him by providing code that lets users enter test scores into the doubly linked list and find the maximum score efficiently.

Input Format

The first line consists of an integer N, representing the number of elements to be

initially inserted into the doubly linked list.

The second line consists of N space-separated integers, denoting the score to be inserted.

Output Format

The output prints an integer, representing the highest score present in the list.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 4
89 71 2 70
Output: 89

Answer

```
// You are using GCC
#include <stdio.h>

#include <stdlib.h>


struct Node {

   int score;

   struct Node* next;

   struct Node* prev;

};


struct Node* createNode(int score) {

   struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));

   newNode->score = score;
```

```
    newNode->next = NULL;
    newNode->prev = NULL;
    return newNode;
}

void append(struct Node** head, int score) {
    struct Node* newNode = createNode(score);
    if (*head == NULL) {
        *head = newNode;
    } else {
        struct Node* temp = *head;
        while (temp->next != NULL) {
            temp = temp->next;
        }
        temp->next = newNode;
        newNode->prev = temp;
    }
}

int findMaxScore(struct Node* head) {
    if (head == NULL) {
        return -1;
```

```
    }

    int maxScore = head->score;
    struct Node* temp = head;

    while (temp != NULL) {
        if (temp->score > maxScore) {
            maxScore = temp->score;
        }
        temp = temp->next;
    }

    return maxScore;
}

void freeList(struct Node* head) {
    struct Node* current = head;
    while (current != NULL) {
        struct Node* next = current->next;
        free(current);
        current = next;
    }
}
```

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

    struct Node* head = NULL;

    for (int i = 0; i < N; i++) {
        int score;
        scanf("%d", &score);
        append(&head, score);
    }
    int maxScore = findMaxScore(head);
    printf("%d\n", maxScore);

    freeList(head);

    return 0;
}
```

Status : Correct **Marks : 10/10**

Rajalakshmi Engineering College

Name: Aashira Fathima
Email: 241501004@rajalakshmi.edu.in
Roll no: 241501004
Phone: 9600884785
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 2_PAH

Attempt : 1
Total Mark : 50
Marks Obtained : 50

Section 1 : Coding

1. Problem Statement

Tom is a software developer working on a project where he has to check if a doubly linked list is a palindrome. He needs to write a program to solve this problem. Write a program to help Tom check if a given doubly linked list is a palindrome or not.

Input Format

The first line consists of an integer N, representing the number of elements in the linked list.

The second line consists of N space-separated integers representing the linked list elements.

Output Format

The first line displays the space-separated integers, representing the doubly

linked list.

The second line displays one of the following:

1. If the doubly linked list is a palindrome, print "The doubly linked list is a palindrome".
2. If the doubly linked list is not a palindrome, print "The doubly linked list is not a palindrome".

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 5
1 2 3 2 1
Output: 1 2 3 2 1
The doubly linked list is a palindrome

Answer

```
#include <stdio.h>

#include <stdlib.h>

struct Node {

   int data;

   struct Node* next;

   struct Node* prev;

};


struct Node* createNode(int data) {

   struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));

   newNode->data = data;
```

```
    newNode->next = NULL;
    newNode->prev = NULL;
    return newNode;
}

int isPalindrome(struct Node* head) {
    struct Node* left = head;
    struct Node* right = head;

    while (right->next != NULL) {
        right = right->next;
    }

    while (left != NULL && right != NULL && left != right && left->prev != right) {
        if (left->data != right->data) {
            return 0;
        }
        left = left->next;
        right = right->prev;
    }
    return 1;
}
```

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

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

    struct Node* head = NULL;
    struct Node* tail = NULL;

    for (int i = 0; i < N; i++) {
        int value;
        scanf("%d", &value);
        struct Node* newNode = createNode(value);
        if (head == NULL) {
            head = newNode;
            tail = newNode;
```

```
    } else {
        tail->next = newNode;
        newNode->prev = tail;
        tail = newNode;
    }
}

printList(head);

if (isPalindrome(head)) {
    printf("The doubly linked list is a palindrome\n");
} else {
    printf("The doubly linked list is not a palindrome\n");
}

struct Node* current = head;
while (current != NULL) {
    struct Node* next = current->next;
    free(current);
    current = next;
}

return 0;
```

}

Status : Correct ***Marks : 10/10***

2. Problem Statement

Bala is a student learning about the doubly linked list and its functionalities. He came across a problem where he wanted to create a doubly linked list by appending elements to the front of the list.

After populating the list, he wanted to delete the node at the given position from the beginning. Write a suitable code to help Bala.

Input Format

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

The second line contains N integers separated by a space, the data values of the nodes in the doubly linked list.

The third line contains an integer X, the position of the node to be deleted from the doubly linked list.

Output Format

The first line of output displays the original elements of the doubly linked list, separated by a space.

The second line prints the updated list after deleting the node at the given position X from the beginning.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
10 20 30 40 50
2

Output: 50 40 30 20 10
50 30 20 10

Answer

```
#include <stdio.h>

#include <stdlib.h>


struct Node {

   int data;

   struct Node* next;

   struct Node* prev;

};


struct Node* createNode(int data) {

   struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));

   newNode->data = data;

   newNode->next = NULL;

   newNode->prev = NULL;

   return newNode;

}


void appendToFront(struct Node** head, int data) {

   struct Node* newNode = createNode(data);

   if (*head == NULL) {

      *head = newNode;
```

```
    } else {
        newNode->next = *head;
        (*head)->prev = newNode;
        *head = newNode;
    }
}

void deleteNodeAtPosition(struct Node** head, int position) {
    if (*head == NULL || position < 1) {
        return;
    }

    struct Node* temp = *head;

    for (int i = 1; temp != NULL && i < position; i++) {
        temp = temp->next;
    }
    if (temp == NULL) {
        return;
    }

    if (temp->prev != NULL) {
        temp->prev->next = temp->next;
```

```
    } else {
        *head = temp->next;
    }

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

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

```
    struct Node* head = NULL;
    for (int i = 0; i < N; i++) {
        int value;
        scanf("%d", &value);
        appendToFront(&head, value);
    }
    int X;
    scanf("%d", &X);
    printList(head);
    deleteNodeAtPosition(&head, X);
    printList(head);
    struct Node* current = head;
    while (current != NULL) {
        struct Node* next = current->next;
        free(current);
        current = next;
    }

    return 0;
}
```

Status : Correct **Marks : 10/10**

3. Problem Statement

Rohan is a software developer who is working on an application that processes data stored in a Doubly Linked List. He needs to implement a feature that finds and prints the middle element(s) of the list. If the list contains an odd number of elements, the middle element should be printed. If the list contains an even number of elements, the two middle elements should be printed.

Help Rohan by writing a program that reads a list of numbers, prints the list, and then prints the middle element(s) based on the number of elements in the list.

Input Format

The first line of the input consists of an integer n the number of elements in the doubly linked list.

The second line consists of n space-separated integers representing the elements of the list.

Output Format

The first line prints the elements of the list separated by space. (There is an extra space at the end of this line.)

The second line prints the middle element(s) based on the number of elements.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
20 52 40 16 18

Output: 20 52 40 16 18
40

Answer

```
#include <stdio.h>

#include <stdlib.h>
```

```
struct Node {
    int data;
    struct Node* next;
    struct Node* prev;
};
struct Node* createNode(int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = NULL;
    newNode->prev = NULL;
    return newNode;
}

void append(struct Node** head, int data) {
    struct Node* newNode = createNode(data);
    if (*head == NULL) {
        *head = newNode;
    } else {
        struct Node* temp = *head;
        while (temp->next != NULL) {
            temp = temp->next;
        }
```

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

void printMiddle(struct Node* head, int n) {
    if (n % 2 == 1) {
        struct Node* temp = head;
        for (int i = 0; i < n / 2; i++) {
            temp = temp->next;
        }
        printf("%d\n", temp->data);
    } else {
```

```
        struct Node* temp = head;
        for (int i = 0; i < n / 2 - 1; i++) {
            temp = temp->next;
        }
        printf("%d %d\n", temp->data, temp->next->data);
    }
}

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

    struct Node* head = NULL;

    for (int i = 0; i < n; i++) {
        int value;
        scanf("%d", &value);
        append(&head, value);
    }

    printList(head);
    printMiddle(head, n);

    struct Node* current = head;
    while (current != NULL) {
```

```
        struct Node* next = current->next;
        free(current);
        current = next;
    }

    return 0;
}
```

Status : Correct *Marks : 10/10*

4. Problem Statement

Riya is developing a contact management system where recently added contacts should appear first. She decides to use a doubly linked list to store contact IDs in the order they are added. Initially, new contacts are inserted at the front of the list. However, sometimes she needs to insert a new contact at a specific position in the list based on priority.

Help Riya implement this system by performing the following operations:

Insert contact IDs at the front of the list as they are added.Insert a new contact at a given position in the list.

Input Format

The first line of input consists of an integer N, representing the initial size of the linked list.

The second line consists of N space-separated integers, representing the values of the linked list to be inserted at the front.

The third line consists of an integer position, representing the position at which the new value should be inserted (position starts from 1).

The fourth line consists of integer data, representing the new value to be inserted.

Output Format

The first line of output prints the original list after inserting initial elements to the front.

The second line prints the updated linked list after inserting the element at the specified position.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 4
10 20 30 40
3
25
Output: 40 30 20 10
40 30 25 20 10

Answer

```
// You are using GCC
#include <stdio.h>

#include <stdlib.h>

struct Node {

    int data;

    struct Node* next;

    struct Node* prev;

};

struct Node* createNode(int data) {

    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));

    newNode->data = data;
```

```
    newNode->next = NULL;
    newNode->prev = NULL;
    return newNode;
}
void insertAtFront(struct Node** head, int data) {
    struct Node* newNode = createNode(data);
    if (*head == NULL) {
        *head = newNode;
    } else {
        newNode->next = *head;
        (*head)->prev = newNode;
        *head = newNode;
    }
}
void insertAtPosition(struct Node** head, int position, int data) {
    struct Node* newNode = createNode(data);
    if (position == 1) {
        insertAtFront(head, data);
        return;
    }
```

```
    struct Node* temp = *head;
    for (int i = 1; temp != NULL && i < position - 1; i++) {
        temp = temp->next;
    }

    if (temp == NULL) {
        return;
    }

    newNode->next = temp->next;
    newNode->prev = temp;

    if (temp->next != NULL) {
        temp->next->prev = newNode;
    }
    temp->next = newNode;
}
void printList(struct Node* head) {
    struct Node* temp = head;
    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->next;
    }
```

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

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

    struct Node* head = NULL;
    for (int i = 0; i < N; i++) {
        int value;
        scanf("%d", &value);
        insertAtFront(&head, value);
    }
    int position, data;
    scanf("%d", &position);
    scanf("%d", &data);
    printList(head);
    insertAtPosition(&head, position, data);
    printList(head);
    struct Node* current = head;
    while (current != NULL) {
        struct Node* next = current->next;
        free(current);
```

```
        current = next;
    }

    return 0;
}
```

Status : Correct ***Marks : 10/10***

5. Problem Statement

Pranav wants to clockwise rotate a doubly linked list by a specified number of positions. He needs your help to implement a program to achieve this. Given a doubly linked list and an integer representing the number of positions to rotate, write a program to rotate the list clockwise.

Input Format

The first line of input consists of an integer n, representing the number of elements in the linked list.

The second line consists of n space-separated linked list elements.

The third line consists of an integer k, representing the number of places to rotate the list.

Output Format

The output displays the elements of the doubly linked list after rotating it by k positions.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 5
1 2 3 4 5
1

Output: 5 1 2 3 4

Answer

```
// You are using GCC
#include <stdio.h>

#include <stdlib.h>

struct Node {

    int data;

    struct Node* next;

    struct Node* prev;

};

struct Node* createNode(int data) {

    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));

    newNode->data = data;

    newNode->next = NULL;

    newNode->prev = NULL;

    return newNode;

}


void append(struct Node** head, int data) {

    struct Node* newNode = createNode(data);

    if (*head == NULL) {

        *head = newNode;

    } else {
```

```
        struct Node* temp = *head;
        while (temp->next != NULL) {
            temp = temp->next;
        }
        temp->next = newNode;
        newNode->prev = temp;
    }
}
void rotateClockwise(struct Node** head, int k) {
    if (*head == NULL || k == 0) {
        return;
    }

    struct Node* current = *head;
    int length = 1;
    while (current->next != NULL) {
        current = current->next;
        length++;
    }
    k = k % length;
    if (k == 0) {
```

```
        return;
    }
    struct Node* newTail = *head;
    for (int i = 1; i < length - k; i++) {
        newTail = newTail->next;
    }

    struct Node* newHead = newTail->next;

    newTail->next = NULL;
    newHead->prev = NULL;
    current->next = *head;
    (*head)->prev = current;
    *head = newHead;
}
void printList(struct Node* head) {
    struct Node* temp = head;
    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->next;
    }
    printf("\n");
```

```
}

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

    struct Node* head = NULL;
    for (int i = 0; i < n; i++) {
        int value;
        scanf("%d", &value);
        append(&head, value);
    }

    int k;
    scanf("%d", &k);

    rotateClockwise(&head, k);
    printList(head);
    struct Node* current = head;
    while (current != NULL) {
        struct Node* next = current->next;
        free(current);
        current = next;
    }

    return 0;
```

}

Status : Correct *Marks : 10/10*