Rajalakshmi Engineering College

Name: Prithika S
Email: 240701400@rajalakshmi.edu.in
Roll no: 240701400
Phone: 9790212894
Branch: REC
Department: I CSE FD
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 2_PAH

Attempt : 2
Total Mark : 50
Marks Obtained : 50

Section 1 : Coding

1. 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>

// Define the node structure
typedef struct Node {
   int data;
   struct Node* prev;
   struct Node* next;
} Node;
```

```
// Function to create a new node
Node* createNode(int data) {
   Node* newNode = (Node*)malloc(sizeof(Node));
   newNode->data = data;
   newNode->prev = NULL;
   newNode->next = NULL;
   return newNode;
}

// Insert at the front of the list
void insertFront(Node** head, int data) {
   Node* newNode = createNode(data);
   newNode->next = *head;
   if (*head != NULL)
      (*head)->prev = newNode;
   *head = newNode;
}

// Insert at a specific position (1-indexed)
void insertAtPosition(Node** head, int position, int data) {
   if (position == 1) {
      insertFront(head, data);
      return;
   }

   Node* newNode = createNode(data);
   Node* current = *head;
   for (int i = 1; i < position - 1 && current != NULL; i++) {
      current = current->next;
   }

   if (current == NULL) return;

   newNode->next = current->next;
   newNode->prev = current;

   if (current->next != NULL)
      current->next->prev = newNode;

   current->next = newNode;
}
```

```
// Print the list
void printList(Node* head) {
   Node* current = head;
   while (current != NULL) {
      printf("%d ", current->data);
      current = current->next;
   }
   printf("\n");
}

// Main function
int main() {
   int N, position, data;
   scanf("%d", &N);

   int elements[N];
   for (int i = 0; i < N; i++) {
      scanf("%d", &elements[i]);
   }

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

   Node* head = NULL;

   // Insert initial elements at front
   for (int i = 0; i < N; i++) {
      insertFront(&head, elements[i]);
   }

   // Print original list
   printList(head);

   // Insert at given position
   insertAtPosition(&head, position, data);

   // Print updated list
   printList(head);

   return 0;
}
```

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

2. 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>

// Define node structure
struct Node {
    int data;
    struct Node* prev;
    struct Node* next;
};

// Create a new node
struct Node* createNode(int data) {
    struct Node* new_node = (struct Node*)malloc(sizeof(struct Node));
    new_node->data = data;
    new_node->prev = NULL;
    new_node->next = NULL;
    return new_node;
}

// Append a node at the end
void append(struct Node** head_ref, int data) {
    struct Node* new_node = createNode(data);
    if (*head_ref == NULL) {
        *head_ref = new_node;
        return;
    }

    struct Node* temp = *head_ref;
    while (temp->next != NULL)
        temp = temp->next;

    temp->next = new_node;
    new_node->prev = temp;
}

// Print the list
void printList(struct Node* head) {
    struct Node* temp = head;
    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->next;
    }
}

// Function to rotate the list clockwise by k nodes
void rotateClockwise(struct Node** head_ref, int k, int n) {
    if (*head_ref == NULL || k == 0 || k >= n)
```

```
        return;

    struct Node* tail = *head_ref;
    // Move to the last node
    while (tail->next != NULL)
        tail = tail->next;

    // Find the new head after rotation
    struct Node* new_tail = tail;
    for (int i = 0; i < k; i++) {
        new_tail = new_tail->prev;
    }

    struct Node* new_head = new_tail->next;

    // Break the list and adjust pointers
    new_tail->next = NULL;
    new_head->prev = NULL;

    tail->next = *head_ref;
    (*head_ref)->prev = tail;

    *head_ref = new_head;
}

int main() {
    int n, k, value;
    struct Node* head = NULL;

    // Input number of elements
    scanf("%d", &n);

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

    // Input rotation count
    scanf("%d", &k);

    // Perform clockwise rotation
    rotateClockwise(&head, k, n);

    // Print rotated list
    printList(head);
    printf("\n");

    return 0;
}
```

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

3. 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

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

// Define node structure
struct Node {
   int data;
   struct Node* prev;
   struct Node* next;
};

// Function to push a node at the front
void pushFront(struct Node** head_ref, int new_data) {
   struct Node* new_node = (struct Node*)malloc(sizeof(struct Node));
   new_node->data = new_data;
   new_node->prev = NULL;
   new_node->next = *head_ref;

   if (*head_ref != NULL)
      (*head_ref)->prev = new_node;

   *head_ref = new_node;
}

// Function to delete node at given position
void deleteAtPosition(struct Node** head_ref, int pos) {
   if (*head_ref == NULL || pos <= 0)
      return;

   struct Node* temp = *head_ref;
   int count = 1;
```

```
    // If head is to be deleted
    if (pos == 1) {
        *head_ref = temp->next;
        if (*head_ref != NULL)
            (*head_ref)->prev = NULL;
        free(temp);
        return;
    }

    // Traverse to the position
    while (temp != NULL && count < pos) {
        temp = temp->next;
        count++;
    }

    if (temp == NULL)
        return;

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

    free(temp);
}

// Function to print the list
void printList(struct Node* node) {
    while (node != NULL) {
        printf("%d ", node->data);
        node = node->next;
    }
}

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

    int i, data;
    struct Node* head = NULL;

    for (i = 0; i < N; i++) {
        scanf("%d", &data);
        pushFront(&head, data);
    }

    scanf("%d", &X);

    // Print original list
    printList(head);
    printf(" ");

    // Delete the node at position X
    deleteAtPosition(&head, X);

    // Print updated list
    printList(head);
    printf("\n");

    return 0;
}
```

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

4. 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

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

// Define node structure
struct Node {
    int data;
    struct Node* prev;
    struct Node* next;
};

// Function to create a new node
struct Node* createNode(int data) {
    struct Node* new_node = (struct Node*)malloc(sizeof(struct Node));
    new_node->data = data;
    new_node->prev = NULL;
    new_node->next = NULL;
    return new_node;
}

// Function to append a node at the end
void append(struct Node** head_ref, int data) {
    struct Node* new_node = createNode(data);
    if (*head_ref == NULL) {
        *head_ref = new_node;
        return;
    }

    struct Node* temp = *head_ref;
    while (temp->next != NULL)
        temp = temp->next;

    temp->next = new_node;
    new_node->prev = temp;
}

// Function to print the list
void printList(struct Node* head) {
    struct Node* temp = head;
    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->next;
    }
}

// Function to find and print the middle element(s)
void printMiddle(struct Node* head, int n) {
    struct Node* temp = head;
    int i;

    if (n % 2 == 1) {
        // Move to middle element
        for (i = 0; i < n / 2; i++)
            temp = temp->next;
        printf("%d\n", temp->data);
    } else {
        // Move to first middle element
        for (i = 0; i < (n / 2) - 1; i++)
            temp = temp->next;
        printf("%d %d\n", temp->data, temp->next->data);
    }
}

int main() {
    int n, i, value;
    struct Node* head = NULL;

    // Input number of elements
    scanf("%d", &n);

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

    // Print the list
    printList(head);
    printf(" ");

    // Print the middle element(s)
    printMiddle(head, n);

    return 0;
}
```

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

5. 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

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

// Define node structure
struct Node {
   int data;
   struct Node* prev;
   struct Node* next;
};

// Function to create a new node
struct Node* createNode(int data) {
   struct Node* new_node = (struct Node*)malloc(sizeof(struct Node));
   new_node->data = data;
   new_node->prev = NULL;
   new_node->next = NULL;
   return new_node;
}
```

```
// Function to append node at the end
void append(struct Node** head_ref, int data) {
    struct Node* new_node = createNode(data);
    if (*head_ref == NULL) {
        *head_ref = new_node;
        return;
    }

    struct Node* temp = *head_ref;
    while (temp->next != NULL)
        temp = temp->next;

    temp->next = new_node;
    new_node->prev = temp;
}

// Function to print the list
void printList(struct Node* head) {
    struct Node* temp = head;
    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->next;
    }
}

// Function to check if the list is a palindrome
bool isPalindrome(struct Node* head) {
    if (head == NULL)
        return true;

    struct Node* left = head;
    struct Node* right = head;

    // Move to the end of the list
    while (right->next != NULL)
        right = right->next;

    // Compare from both ends
    while (left != right && right->next != left) {
        if (left->data != right->data)
            return false;
        left = left->next;
        right = right->prev;
    }

    return true;
}

int main() {
    int N, i, value;
    struct Node* head = NULL;

    // Read number of elements
    scanf("%d", &N);

    // Read elements and build list
    for (i = 0; i < N; i++) {
        scanf("%d", &value);
        append(&head, value);
    }

    // Print the list
    printList(head);
    printf(" ");

    // Check and print if palindrome
    if (isPalindrome(head))
        printf("The doubly linked list is a palindrome\n");
    else
        printf("The doubly linked list is not a palindrome\n");

    return 0;
}
```

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