Rajalakshmi Engineering College

Name: KEERTHI PRIYA T
Email: 240701258@rajalakshmi.edu.in
Roll no: 2116240701258
Phone: 7397397221
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

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

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

2116240701258

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

// Node structure for doubly linked list
typedef struct Node {
   int data;
   struct Node* prev;
   struct Node* next;
} Node;

// 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;
}
```

211624701258

```c
// Insert node at the end of the list
void insertEnd(Node** head, int data) {
   Node* newNode = createNode(data);
   if (*head == NULL) {
      *head = newNode;
      return;
   }
   Node* temp = *head;
   while (temp->next != NULL)
      temp = temp->next;
   temp->next = newNode;
   newNode->prev = temp;
}

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

// Print middle element(s)
void printMiddle(Node* head, int n) {
   Node* temp = head;
   int count = 0;
   while (count < (n / 2)) {
      temp = temp->next;
      count++;
   }

   if (n % 2 == 1) {
      // Odd number of elements: print single middle
      printf("%d", temp->data);
   } else {
      // Even number of elements: print two middle elements
      printf("%d %d", temp->prev->data, temp->data);
   }
}
```

2116240701258

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

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

   // Read n elements and insert into list
   for (int i = 0; i < n; i++) {
      scanf("%d", &value);
      insertEnd(&head, value);
   }

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

   // Print middle element(s)
   printMiddle(head, n);
   printf("\n");

   return 0;
}
```

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

2. 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.

2116240701258

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>
#include <stdbool.h>

// Doubly linked list node structure
typedef struct Node {
    int data;
    struct Node* prev;
    struct Node* next;
} Node;

// 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;
```

2116240701258

```
}

// Insert a node at the end
void insertEnd(Node** head, int data) {
   Node* newNode = createNode(data);
   if (*head == NULL) {
      *head = newNode;
      return;
   }
   Node* temp = *head;
   while (temp->next != NULL)
      temp = temp->next;
   temp->next = newNode;
   newNode->prev = temp;
}

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

// Check if list is a palindrome
bool isPalindrome(Node* head) {
   if (head == NULL) return true;

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

   // Move right to the last node
   while (right->next != NULL) {
      right = right->next;
   }

   // Compare data from both ends
   while (left != NULL && right != NULL && left != right && left->prev != right) {
      if (left->data != right->data)
         return false;
      left = left->next;
```
2116240701258

```
        right = right->prev;
    }
    return true;
}

int main() {
    int n, val;
    Node* head = NULL;

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

    // Read and insert elements
    for (int i = 0; i < n; i++) {
        scanf("%d", &val);
        insertEnd(&head, val);
    }

    // Print list
    printList(head);

    // Check and print palindrome status
    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***

3. 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.

2116240701258

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>

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

// Create a new node
Node* createNode(int data) {
   Node* newNode = (Node*) malloc(sizeof(Node));
   newNode->data = data;
   newNode->prev = NULL;
```

2116240701258

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

// Insert node at the end
void insertEnd(Node** head, int data) {
    Node* newNode = createNode(data);
    if (*head == NULL) {
        *head = newNode;
        return;
    }
    Node* temp = *head;
    while (temp->next != NULL)
        temp = temp->next;
    temp->next = newNode;
    newNode->prev = temp;
}

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

// Rotate list clockwise by k positions
void rotateClockwise(Node** head, int k, int n) {
    if (*head == NULL || k == 0 || k >= n)
        return;

    // Step 1: Go to the last node
    Node* tail = *head;
    while (tail->next != NULL)
        tail = tail->next;

    // Step 2: Go k steps back from tail
    Node* newTail = tail;
    for (int i = 0; i < k; i++)
        newTail = newTail->prev;
```

2116240701258

```
    Node* newHead = newTail->next;

    // Break and relink pointers
    newTail->next = NULL;
    newHead->prev = NULL;
    tail->next = *head;
    (*head)->prev = tail;
    *head = newHead;
}

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

    // Read number of nodes
    scanf("%d", &n);

    // Read list elements
    for (int i = 0; i < n; i++) {
        scanf("%d", &val);
        insertEnd(&head, val);
    }

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

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

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

    return 0;
}
```

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

4. Problem Statement

Bala is a student learning about the doubly linked list and its

2116240701258

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>

// Node structure
```

2116240701258

```
typedef struct Node {
   int data;
   struct Node* prev;
   struct Node* next;
} Node;

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

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

// Delete node at position (1-based)
void deleteAtPosition(Node** head, int pos) {
   if (*head == NULL || pos < 1)
      return;

   Node* temp = *head;

   // Traverse to the node to delete
```

2116240701258

```
    for (int i = 1; i < pos && temp != NULL; i++) {
        temp = temp->next;
    }

    if (temp == NULL)
        return;

    // If node to delete is head
    if (temp == *head) {
        *head = temp->next;
        if (*head)
            (*head)->prev = NULL;
    } else {
        if (temp->prev)
            temp->prev->next = temp->next;
        if (temp->next)
            temp->next->prev = temp->prev;
    }

    free(temp);
}

int main() {
    int n, x, val;
    Node* head = NULL;

    // Read number of nodes
    scanf("%d", &n);

    // Read values and insert at front
    for (int i = 0; i < n; i++) {
        scanf("%d", &val);
        insertFront(&head, val);
    }

    // Read position to delete
    scanf("%d", &x);

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

2116240701258

```
    // Delete node at position x
    deleteAtPosition(&head, x);

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

    return 0;
}
```

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

5. 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.

2116240701258

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>

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

// 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 front
void insertFront(Node** head, int data) {
   Node* newNode = createNode(data);
   if (*head != NULL) {
```

2116240701258

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

// Insert at a specific position (1-based)
void insertAtPosition(Node** head, int pos, int data) {
    Node* newNode = createNode(data);

    if (pos == 1) {
        insertFront(head, data);
        return;
    }

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

    if (temp == NULL || temp->next == NULL) {
        // Insert at end
        temp->next = newNode;
        newNode->prev = temp;
    } else {
        newNode->next = temp->next;
        newNode->prev = temp;
        temp->next->prev = newNode;
        temp->next = newNode;
    }
}

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

int main() {
```
2116240701258

```
    int n, pos, val;
    Node* head = NULL;

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

    // Read and insert elements at front
    for (int i = 0; i < n; i++) {
        scanf("%d", &val);
        insertFront(&head, val);
    }

    // Read position and new data
    scanf("%d", &pos);
    scanf("%d", &val);

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

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

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

    return 0;
}
```

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

2116240701258