Rajalakshmi Engineering College

Name: Tejeshwaran P
Email: 241801292@rajalakshmi.edu.in
Roll no: 241801292
Phone: 6383048813
Branch: REC
Department: l AI & DS AF
Batch: 2028
Degree: B.E - AI & DS

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

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
2
10 20 30 40 50

Output: 50 40 30 20 10
50 30 20 10

Answer

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

// Node structure for doubly linked list struct Node
{
   int data;  // Data of the node
   struct Node* next;  // Pointer to the next node                    struct Node*
prev;  // Pointer to the previous node
};
```

```
// Function to insert a node at the front of the doubly linked list void
insertFront(struct Node** head, int val) {
   struct Node* newNode = (struct
Node*)malloc(sizeof(struct Node));     newNode-                        >data = val;
newNode->next = *head;     newNode->prev =                             NULL;

   // If the list is not empty, update the previous                    pointer of
the old head
   if (*head != NULL) {        (*head)->prev =                         newNode;

    }
   *head = newNode;  // Make the new node the head of the list

}
// Function to delete a node at a given position from the beginning void
deleteAtPosition(struct Node** head, int position) {
   if (*head == NULL) {
      return;  // If the list is empty, return
   }

   struct Node* temp = *head;

   // Traverse to the desired position     for (int i = 1;             temp !=
NULL && i < position; i++) {        temp = temp-                       >next;
   }

   // If the position is out of bounds     if
(temp == NULL) {
      return;     }

   // If the node to be deleted is the head     if
(temp == *head) {        *head = temp->next;        if
(*head != NULL) {            (*head)->prev = NULL;
      }
      free(temp);
      return;     }
   // If the node to be deleted is not the head     if (temp->next != NULL) {         temp-
>next->prev = temp->prev;
    }
   if (temp->prev != NULL) {         temp->prev->next
= temp->next;     free(temp);  // Free the memory
    }
of the node
}

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

int main() {     int N, X;
   scanf("%d", &N);  // Read the number of                          elements to

be inserted     struct Node* head = NULL;

   // Insert the elements into the doubly linked list
   for (int i = 0; i < N; i++) {        int value;        scanf("%d",
&value);
       insertFront(&head, value);  // Insert at the                              front
   }

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

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

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

```
    // Print the                                    updated list
    printf(" ");

     return 0;
}
```

printList(head); *Status :*
Correct*Marks : 10/10*