Rajalakshmi Engineering College

Name: GOGUL ANAND.P
Email: 240801081@rajalakshmi.edu.in
Roll no: 2116240801081
Phone: 8248075810
Branch: REC
Department: I ECE FA
Batch: 2028
Degree: B.E - ECE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 2_COD_Question 2

Attempt : 1
Total Mark : 10
Marks Obtained : 0

Section 1 : Coding

1. Problem Statement

Moniksha, a chess coach organizing a tournament, needs a program to manage participant IDs efficiently. The program maintains a doubly linked list of IDs and offers two functions: Append to add IDs as students register, and Print Maximum ID to identify the highest ID for administrative tasks.

This tool streamlines tournament organization, allowing Moniksha to focus on coaching her students effectively.

Input Format

The first line consists of an integer n, representing the number of participant IDs to be added.

The second line consists of n space-separated integers representing the participant IDs.

Output Format

The output displays a single integer, representing the maximum participant ID.

If the list is empty, the output prints "Empty list!".

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 3
163 137 155
Output: 163

Answer

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

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

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

void append(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;

}


void pushFront(Node** head, int data) {
    Node* newNode = createNode(data);
    *head = newNode;
}

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

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

    Node* evenHead = NULL;
    Node* oddHead = NULL;

    for(int i = 0; i < n; i++) {
        scanf("%d", &val);
        if (val % 2 == 0) {

            pushFront(&evenHead, val);
        } else {


            append(&oddHead, val);
        }
    }

    Node* temp = evenHead;
    if (evenHead == NULL) {
        printList(oddHead);
    } else {
        while (temp->next !=NULL)
            temp = temp->next;
        temp->next = oddHead;
        printList(evenHead);
    }

    return 0;

}
```

Status : Wrong ***Marks : 0/10***