Rajalakshmi Engineering College

Name: DHARSHINI S
Email: 241901026@rajalakshmi.edu.in
Roll no: 241901026
Phone: 7448906612
Branch: REC
Department: l CSE (CS) FA
Batch: 2028
Degree: B.E - CSE (CS)

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 2_MCQ_Updated

Attempt : 1
Total Mark : 20
Marks Obtained : 18

Section 1 : MCQ

1. What is the main advantage of a two-way linked list over a one-way linked list?

Answer

Two-way linked lists allow for traversal in both directions.

Status : Correct   *Marks : 1/1*

2. How do you delete a node from the middle of a doubly linked list?

Answer

All of the mentioned options

Status : Correct   *Marks : 1/1*

3. What is a memory-efficient double-linked list?

Answer

A doubly linked list that uses bitwise AND operator for storing addresses

Status : Correct   *Marks : 1/1*

4. What happens if we insert a node at the beginning of a doubly linked list?

Answer

The previous pointer of the new node is NULL

Status : Correct   *Marks : 1/1*

5. Which of the following information is stored in a doubly-linked list's nodes?

Answer

All of the mentioned options

Status : Correct   *Marks : 1/1*

6. What is the correct way to add a node at the beginning of a doubly linked list?

Answer

void addFirst(int data){   Node* newNode = new Node(data);   newNode->next = head;   if (head != NULL) {   head->prev = newNode;   }   head = newNode;   }

Status : Correct   *Marks : 1/1*

7. Consider the provided pseudo code. How can you initialize an empty two-way linked list?

Define Structure Node

```
    data: Integer
    prev: Pointer to Node
    next: Pointer to Node
End Define

Define Structure TwoWayLinkedList
    head: Pointer to Node
    tail: Pointer to Node
End Define
```

Answer

struct TwoWayLinkedList* list = malloc(sizeof(struct TwoWayLinkedList)); list->head = NULL; list->tail = NULL;

Status : Correct   *Marks : 1/1*

8.   Which code snippet correctly deletes a node with a given value from a doubly linked list?

```
void deleteNode(Node** head_ref, Node* del_node) {
    if (*head_ref == NULL || del_node == NULL) {
        return;
    }
    if (*head_ref == del_node) {
        *head_ref = del_node->next;
    }
    if (del_node->next != NULL) {
        del_node->next->prev = del_node->prev;
    }
    if (del_node->prev != NULL) {
        del_node->prev->next = del_node->next;
    }
    free(del_node);
}
```

Answer

Deletes the node at a given position in a doubly linked list.

Status : Wrong   *Marks : 0/1*

9. Which of the following statements correctly creates a new node for a doubly linked list?

Answer

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

Status : Correct   *Marks : 1/1*

10. Which of the following is false about a doubly linked list?

Answer

Implementing a doubly linked list is easier than singly linked list

Status : Correct   *Marks : 1/1*

11. Which of the following is true about the last node in a doubly linked list?

Answer

Its next pointer is NULL

Status : Correct   *Marks : 1/1*

12. Where Fwd and Bwd represent forward and backward links to the adjacent elements of the list. Which of the following segments of code deletes the node pointed to by X from the doubly linked list, if it is assumed that X points to neither the first nor the last node of the list?

A doubly linked list is declared as

```
struct Node {
    int Value;
    struct Node *Fwd;
    struct Node *Bwd;
);
```

Answer

X->Bwd->Fwd = X->Bwd ; X->Fwd->Bwd = X->Fwd;

13.   What will be the output of the following program?

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

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

int main() {
    struct Node* head = NULL;
    struct Node* tail = NULL;
    for (int i = 0; i < 5; i++) {
        struct Node* temp = (struct Node*)malloc(sizeof(struct Node));
        temp->data = i + 1;
        temp->prev = tail;
        temp->next = NULL;
        if (tail != NULL) {
            tail->next = temp;
        } else {
            head = temp;
        }
        tail = temp;
    }
    struct Node* current = head;
    while (current != NULL) {
        printf("%d ", current->data);
        current = current->next;
    }
    return 0;
}
```

Answer

1 2 3 4 5

14. How many pointers does a node in a doubly linked list have?

Answer

2

Status : Correct   *Marks : 1/1*

15. What will be the output of the following code?

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

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

int main() {
   struct Node* head = NULL;
   struct Node* temp = (struct Node*)malloc(sizeof(struct Node));
   temp->data = 2;
   temp->next = NULL;
   temp->prev = NULL;
   head = temp;
   printf("%d\n", head->data);
   free(temp);
   return 0;
}
```

Answer

2

Status : Correct   *Marks : 1/1*

16. Consider the following function that refers to the head of a Doubly

Linked List as the parameter. Assume that a node of a doubly linked list has the previous pointer as prev and the next pointer as next.

Assume that the reference of the head of the following doubly linked list is passed to the below function 1 <--> 2 <--> 3 <--> 4 <--> 5 <-->6. What should be the modified linked list after the function call?

```
Procedure fun(head_ref: Pointer to Pointer of node)
    temp = NULL
    current = *head_ref

    While current is not NULL
        temp = current->prev
        current->prev = current->next
        current->next = temp
        current = current->prev
    End While

    If temp is not NULL
        *head_ref = temp->prev
    End If
End Procedure
```

Answer

6 <--> 5 <--> 4 <--> 3 <--> 2 <--> 1.

Status : Correct   *Marks : 1/1*

17.   What will be the effect of setting the prev pointer of a node to NULL in a doubly linked list?

Answer

The node will become the new head

Status : Correct   *Marks : 1/1*

18.   What does the following code snippet do?

```
struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
newNode->data = value;
newNode->next = NULL;
newNode->prev = NULL;
```

Answer

Creates a new node and initializes its data to 'value'

Status : Correct   *Marks : 1/1*

19.   Which pointer helps in traversing a doubly linked list in reverse order?

Answer

prev

Status : Correct   *Marks : 1/1*

20.   How do you reverse a doubly linked list?

Answer

By swapping the next and previous pointers of each node

Status : Correct   *Marks : 1/1*

Rajalakshmi Engineering College

Name: DHARSHINI S
Email: 241901026@rajalakshmi.edu.in
Roll no: 241901026
Phone: 7448906612
Branch: REC
Department: l CSE (CS) FA
Batch: 2028
Degree: B.E - CSE (CS)

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 2_COD_Question 1

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Your task is to create a program to manage a playlist of items. Each item is represented as a character, and you need to implement the following operations on the playlist.

Here are the main functionalities of the program:

Insert Item: The program should allow users to add items to the front and end of the playlist. Items are represented as characters.Display Playlist: The program should display the playlist containing the items that were added.

To implement this program, a doubly linked list data structure should be used, where each node contains an item character.

Input Format

The input consists of a sequence of space-separated characters, representing the items to be inserted into the doubly linked list.

The input is terminated by entering - (hyphen).

Output Format

The first line of output prints "Forward Playlist: " followed by the linked list after inserting the items at the end.

The second line prints "Backward Playlist: " followed by the linked list after inserting the items at the front.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: a b c -
Output: Forward Playlist: a b c
Backward Playlist: c b a

Answer

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

struct Node {
char item;
   struct Node* next;
   struct Node* prev;
};
// You are using GCC
void insertAtEnd(struct Node** head, char item) {
   struct Node* newn=(struct Node*)malloc(sizeof(struct Node));
   newn->item=item;
   newn->next=NULL;
   newn->prev=NULL;
   if(*head==NULL){
     *head=newn;
}
   else{
```

```c
        struct Node* temp=*head;
        while(temp->next!=NULL){
            temp=temp->next;
        }
        temp->next=newn;
        newn->prev=temp;
    }
}
void displayForward(struct Node* head) {
    struct Node* temp=head;
    while(temp){
        printf("%c ",temp->item);
        temp=temp->next;
    }

}

void displayBackward(struct Node* tail) {
    struct Node* temp=tail;
    while(temp->next){
        temp=temp->next;
    }
    while(temp){
        printf("%c ",temp->item);
        temp=temp->prev;
    }
    printf("\n");

}

void freePlaylist(struct Node* head) {
    struct Node* temp=head;
    free(temp);

}
int main() {
    struct Node* playlist = NULL;
    char item;

    while (1) {
        scanf(" %c", &item);
        if (item == '-') {
```

```c
        break;
    }
    insertAtEnd(&playlist, item);
}

struct Node* tail = playlist;
while (tail->next != NULL) {
    tail = tail->next;
}

printf("Forward Playlist: ");
displayForward(playlist);

printf("Backward Playlist: ");
displayBackward(tail);

freePlaylist(playlist);

return 0;
}
```

Status : Correct   *Marks : 10/10*

Rajalakshmi Engineering College

Name: DHARSHINI S
Email: 241901026@rajalakshmi.edu.in
Roll no: 241901026
Phone: 7448906612
Branch: REC
Department: l CSE (CS) FA
Batch: 2028
Degree: B.E - CSE (CS)

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 : 10

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

```c
// You are using GCC
#include<stdio.h>
#include<stdlib.h>
struct node{
    int data;
    struct node* next;
    struct node* prev;

};
void insertatend(struct node** head,struct node** tail,int data)
{
    struct node* newn=(struct node*)malloc(sizeof(struct node));
    newn->data=data;
    newn->next=NULL;
    newn->prev=*tail;
    if(*tail==NULL)
    {
        *head=newn;
        *tail=newn;
    }
    else
    {
        (*tail)->next=newn;
        *tail=newn;
    }
```

```c
}
int max(struct node* head)
{
    if(head==NULL)
    {
        return -1;
    }
    int maxid = head->data;
    struct node* temp = head->next;
    while(temp)
    {
        if(temp->data > maxid)
        {
            maxid=temp->data;
        }
        temp=temp->next;
    }
    return maxid;
}
void freelist(struct node* head)
{
    struct node* temp;
    while(head)
    {
        temp=head;
        head=head->next;
        free(temp);

    }
}
int main()
{
    struct node* head=NULL;
    struct node* tail=NULL;
    int n;
    scanf("%d",&n);
    if(n==0)
    {
        printf("Empty list!\n");
        return 0;
    }
    for(int i=0;i<n;i++)
```

```c
    {
        int id;
        scanf("%d",&id);
        insertatend(&head,&tail,id);
    }
    int maxid=max(head);
    if(maxid==-1)
    printf("Empty list!\n");
    else
    printf("%d\n",maxid);
    freelist(head);
    return 0;
}
```

Status : Correct   *Marks : 10/10*

Rajalakshmi Engineering College

Name: DHARSHINI S
Email: 241901026@rajalakshmi.edu.in
Roll no: 241901026
Phone: 7448906612
Branch: REC
Department: l CSE (CS) FA
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 2_COD_Question 3

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Bob is tasked with developing a company's employee record management system. The system needs to maintain a list of employee records using a doubly linked list. Each employee is represented by a unique integer ID.

Help Bob to complete a program that adds employee records at the front, traverses the list, and prints the same for each addition of employees to the list.

Input Format

The first line of input consists of an integer N, representing the number of employees.

The second line consists of N space-separated integers, representing the employee IDs.

Output Format

For each employee ID, the program prints "Node Inserted" followed by the current state of the doubly linked list in the next line, with the data values of each node separated by spaces.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 4
101 102 103 104
Output: Node Inserted
101
Node Inserted
102 101
Node Inserted
103 102 101
Node Inserted
104 103 102 101

Answer

```cpp
#include <iostream>
using namespace std;

struct node {
    int info;
    struct node* prev, * next;
};

struct node* start = NULL;

// You are using GCC
struct node* head=NULL;
void traverse() {
    struct node* temp=head;
    printf("Node Inserted\n");
    while(temp){
    printf("%d ",temp->info);
        temp=temp->next;
    }
```

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

}

void insertAtFront(int data) {
  struct node* newn=(struct node*)malloc(sizeof(struct node));
  newn->info=data;
  newn->prev=NULL;
  newn->next=head;
  head=newn;
}

int main() {
  int n, data;
  cin >> n;
  for (int i = 0; i < n; ++i) {
    cin >> data;
    insertAtFront(data);
    traverse();
  }
  return 0;
}
```

Status : Correct   *Marks : 10/10*

Rajalakshmi Engineering College

Name: DHARSHINI S
Email: 241901026@rajalakshmi.edu.in
Roll no: 241901026
Phone: 7448906612
Branch: REC
Department: l CSE (CS) FA
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 2_COD_Question 4

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Ravi is developing a student registration system for a college. To efficiently store and manage the student IDs, he decides to implement a doubly linked list where each node represents a student's ID.

In this system, each student's ID is stored sequentially, and the system needs to display all registered student IDs in the order they were entered.

Implement a program that creates a doubly linked list, inserts student IDs, and displays them in the same order.

Input Format

The first line contains an integer N the number of student IDs.

The second line contains N space-separated integers representing the student IDs.

Output Format

The output should display the single line containing N space-separated integers representing the student IDs stored in the doubly linked list.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
10 20 30 40 50
Output: 10 20 30 40 50

Answer

```c
// You are using GCC
#include<stdio.h>
#include<stdlib.h>
typedef struct node{
   int data;
   struct node *prev,*next;
}node;
node* cnode(int data){
   node* newn=(node*)malloc(sizeof(node));
   newn->data=data;
   newn->prev=NULL;
   newn->next=NULL;
   return newn;
}
void insert(node** head,int val){
   node* newn=cnode(val);
   if(*head==NULL){
     *head=newn;
   }
   else{
     node* temp=*head;
     while(temp->next!=NULL){
        temp=temp->next;
     }
```

```c
        temp->next=newn;

    }
}
void dis(node* head){
    node* temp=head;
    while(temp!=NULL){
        printf("%d ",temp->data);
        temp=temp->next;
    }
}
int main()
{
    node* head=NULL;
    int n,val;
    scanf("%d",&n);
    for(int i=0;i<n;i++){
        scanf("%d",&val);
        insert(&head,val);
    }
    dis(head);
}
```

Status : Correct   *Marks : 10/10*

Rajalakshmi Engineering College

Name: DHARSHINI S
Email: 241901026@rajalakshmi.edu.in
Roll no: 241901026
Phone: 7448906612
Branch: REC
Department: l CSE (CS) FA
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 2_COD_Question 5

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Ashwin is tasked with developing a simple application to manage a list of items in a shop inventory using a doubly linked list. Each item in the inventory has a unique identification number. The application should allow users to perform the following operations:

Create a List of Items: Initialize the inventory with a given number of items. Each item will be assigned a unique number provided by the user and insert the elements at end of the list.

Delete an Item: Remove an item from the inventory at a specific position.

Display the Inventory: Show the list of items before and after deletion.

If the position provided for deletion is invalid (e.g., out of range), it should

display an error message.

Input Format

The first line contains an integer n, representing the number of items to be initially entered into the inventory.

The second line contains n integers, each representing the unique identification number of an item separated by spaces.

The third line contains an integer p, representing the position of the item to be deleted from the inventory.

Output Format

The first line of output prints "Data entered in the list:" followed by the data values of each node in the doubly linked list before deletion.

If p is an invalid position, the output prints "Invalid position. Try again."

If p is a valid position, the output prints "After deletion the new list:" followed by the data values of each node in the doubly linked list after deletion.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 4
1 2 3 4
5
Output: Data entered in the list:
 node 1 : 1
 node 2 : 2
 node 3 : 3
 node 4 : 4
Invalid position. Try again.

Answer

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

```c
typedef struct node{
    int data;
    struct node *prev,*next;
}node;
node* cnode(int val){
    node* newn=(node*)malloc(sizeof(node));
    newn->data=val;
    newn->prev=NULL;
    newn->next=NULL;
    return newn;
}
void insert(node** head,int data){
    node *newn=cnode(data);
    if(*head==NULL){
        *head=newn;
    }
    else{
        node* temp=*head;
        while(temp->next!=NULL)
        {
            temp=temp->next;
        }
        temp->next=newn;
        newn->prev=temp;
    }
}
void dis(node* head){
    node* temp=head;
    printf("Data entered in the list:\n");
    for(int i=1; temp!=NULL; i++){
        printf("node %d : %d\n",i,temp->data);
        temp=temp->next;
    }
}
void del(node**head,int pos)
{
    node* temp=*head;
    node* trav=*head;
    int i=1;
    while(temp!=NULL&&i<pos){
        temp=temp->next;
        i++;
```

```c
    }
    if(temp==NULL){
        printf("Invalid position. Try again.");
    }
    if(temp->prev==NULL){
        *head=temp->next;
        if(*head!=NULL)
        (*head)->prev=NULL;
    }
    else{
        temp->prev->next=temp->next;
        if(temp->next!=NULL)
            temp->next->prev=temp->prev;
    }
    free(temp);
    printf("After deletion the new list:\n");
    for(int i=1; trav!=NULL; i++){
        printf("node %d : %d\n",i,trav->data);
        trav=trav->next;

    }
}
int main()
{
    node* head=NULL;
    int n,val,pos;
    scanf("%d",&n);
    if(1<n && n<=20){
    for(int i=0; i<n; i++)
        {
            scanf("%d",&val);
            insert(&head,val);
        }
        dis(head);
        scanf("%d",&pos);
        if(1<=pos&&pos<=n){
            del(&head,pos);
        }
        else{
        printf("Invalid position. Try again.\n");
        }
    }
```

```c
    else
    {
        printf("No data found in the list yet.Invalid position, try again.");
    }
}
```

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

Rajalakshmi Engineering College

Name: DHARSHINI S
Email: 241901026@rajalakshmi.edu.in
Roll no: 241901026
Phone: 7448906612
Branch: REC
Department: l CSE (CS) FA
Batch: 2028
Degree: B.E - CSE (CS)

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

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

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

void push(Node** head, node** tail,int value){
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->data = value;
    newNode->next = NULL;

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

```c
        }
    }

    void popBottom(Node** head) {
      if (*head == NULL) {
        return;
      }

      Node* temp = *head;
      *head = (*head)->next;

      if(*head != NULL) {
        (*head)->prev = NULL;
      }

      free(temp);

    }

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

    }

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

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

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

```
    popBottom(&head);

    printStack(head);

    return 0;
}
```

2. Problem Statement

Krishna needs to create a doubly linked list to store and display a sequence of integers. Your task is to help write a program to read a list of integers from input, store them in a doubly linked list, and then display the list.

Input Format

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

The second line of input consists of n space-separated integers.

Output Format

The output prints a single line displaying the integers in the order they were added to the doubly linked list, separated by spaces.

If nothing is added (i.e., the list is empty), it will display "List is empty".

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 5
1 2 3 4 5
Output: 1 2 3 4 5

Answer

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

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

Node* createNode(int data) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    if (!newNode) {
        printf("Memory allocation failed\n");
        exit(1);
    }
    newNode->data = data;
    newNode->prev = NULL;
```

```c
    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;
    newNode->prev = temp;
}

void display(Node* head) {
    if (head == NULL) {
        printf("List is empty\n");
        return;
    }
    Node* temp = head;
    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->next;
    }
    printf("\n");
}

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

    scanf("%d", &n);
    if (n == 0) {
        printf("List is empty\n");
        return 0;
    }

    for (int i = 0; i < n; i++) {
```

```
        scanf("%d", &value);
        append(&head, value);
    }

    display(head);
    return 0;
}
```

Status : Correct   *Marks : 10/10*

3. Problem Statement

Imagine you're managing a store's inventory list, and some products were
accidentally entered multiple times. You need to remove the duplicate
products from the list to ensure each product appears only once.

You have an unsorted doubly linked list of product IDs. Some of these
product IDs may appear more than once, and your goal is to remove any
duplicates.

Input Format

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

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

Output Format

The output prints the final after removing duplicate nodes, separated by a space.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 10
12 12 10 4 8 4 6 4 4 8
Output: 8 4 6 10 12

Answer

```
// You are using GCC
#include<stdio.h>
#include<stdlib.h>
typedef struct Node
{
    struct Node* prev;
    int data;
    struct Node* next;

}node;
node* head=NULL;
void insert(int e)
{
    node* newnode=(node*)malloc(sizeof(node));
    newnode->data=e;
    newnode->prev=NULL;
    newnode->next=head;
    if(head!=NULL)
    {
        head->prev=newnode;
    }
    head=newnode;
}

void removeduplicates()
{
    node* current=head;
    while(current!=NULL)
    {
        node* runner=current->next;
        while(runner!=NULL)
        {
            node* nextnode=runner->next;
            if(runner->data==current->data)
            {
                if(runner->next!=NULL)
                {
                    runner->next->prev=runner->prev;
                }
                if(runner->prev!=NULL)
                {
```

```c
                runner->prev->next=runner->next;
            }
            free(runner);
        }
        runner=nextnode;
    }
    current=current->next;
  }
}
void display()
{
  node* temp=head;
  while(temp!=NULL)
  {
    printf("%d ",temp->data);
    temp=temp->next;
  }
}
int main()
{
  int n,e;
  scanf("%d",&n);
  for(int i=0;i<n;i++)
  {
    scanf("%d",&e);
    insert(e);
  }
  removeduplicates();
  display();
  return 0;
}
```

Status : Correct   *Marks : 10/10*

Rajalakshmi Engineering College

Name: DHARSHINI S
Email: 241901026@rajalakshmi.edu.in
Roll no: 241901026
Phone: 7448906612
Branch: REC
Department: l CSE (CS) FA
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 1_PAH_modified

Attempt : 2
Total Mark : 5
Marks Obtained : 5

Section 1 : Coding

1. Problem Statement

John is working on evaluating polynomials for his math project. He needs to compute the value of a polynomial at a specific point using a singly linked list representation.

Help John by writing a program that takes a polynomial and a value of x as input, and then outputs the computed value of the polynomial.

Example

Input:

2

13

12

11

1

Output:

36

Explanation:

The degree of the polynomial is 2.

Calculate the value of x2: 13 * 12 = 13.

Calculate the value of x1: 12 * 11 = 12.

Calculate the value of x0: 11 * 10 = 11.

Add the values of x2, x1 and x0 together: 13 + 12 + 11 = 36.

Input Format

The first line of input consists of the degree of the polynomial.

The second line consists of the coefficient x2.

The third line consists of the coefficient of x1.

The fourth line consists of the coefficient x0.

The fifth line consists of the value of x, at which the polynomial should be evaluated.

Output Format

The output is the integer value obtained by evaluating the polynomial at the given value of x.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 2

13

12
11
1
Output: 36

Answer

```c
// You are using GCC
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
struct node{
    int data;
    struct node *next;
};struct node *head=NULL,*last=NULL,*ptr,*newnode;
int main()
{int a,b,s=0;
scanf("%d",&a);
for(int i=0;i<a+1;i++)
{newnode=(struct node*)malloc (sizeof(struct node));
    scanf("%d",&newnode->data);
    newnode->next=NULL;
    if(head==NULL)
    {head=newnode;}
    else
    {last->next=newnode;}
    last=newnode;
}
scanf("%d",&b);
int n=a;
ptr=head;
while(ptr!=NULL)
{s+=(pow(b,n--))*(ptr->data);
ptr=ptr->next;}
printf("%d",s);
}
```

Status : Correct   *Marks : 1/1*

2. Problem Statement

Imagine you are managing the backend of an e-commerce platform.

Customers place orders at different times, and the orders are stored in two separate linked lists. The first list holds the orders from morning, and the second list holds the orders from the evening.

Your task is to merge the two lists so that the final list holds all orders in sequence from the morning list followed by the evening orders, in the same order

Input Format

The first line contains an integer n , representing the number of orders in the morning list.

The second line contains n space-separated integers representing the morning orders.

The third line contains an integer m , representing the number of orders in the evening list.

The fourth line contains m space-separated integers representing the evening orders.

Output Format

The output should be a single line containing space-separated integers representing the merged order list, with morning orders followed by evening orders.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 3
101 102 103
2
104 105
Output: 101 102 103 104 105

Answer

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

```c
#include<stdlib.h>
struct node{
    int data;
    struct node*next;};
    struct node *head=NULL,*last=NULL,*newnode,*ptr;
void create(int a)
{
for(int i=0;i<a;i++)
{
newnode=(struct node*)malloc(sizeof(struct node));
scanf("%d",&newnode->data);
newnode->next=NULL;
if(head==NULL)
{head=newnode;}
else
{last->next=newnode;}
last=newnode;
}}
void display()
{ptr=head;
    while(ptr!=NULL)
    {printf("%d ",ptr->data);
        ptr=ptr->next;
    }
}
int main()
{int a,b;
scanf("%d",&a);
create(a);
scanf("%d",&b);
create(b);
display();
return 0;
}
```

Status : Correct   ***Marks : 1/1***

3. Problem Statement

Emily is developing a program to manage a singly linked list. The program should allow users to perform various operations on the linked list, such as

inserting elements at the beginning or end, deleting elements from the beginning or end, inserting before or after a specific value, and deleting elements before or after a specific value. After each operation, the updated linked list should be displayed.

Your task is to help Emily in implementing the same.

Input Format

The first line contains an integer choice, representing the operation to perform:

- For choice 1 to create the linked list. The next lines contain space-separated integers, with -1 indicating the end of input.
- For choice 2 to display the linked list.
- For choice 3 to insert a node at the beginning. The next line contains an integer data representing the value to insert.
- For choice 4 to insert a node at the end. The next line contains an integer data representing the value to insert.
- For choice 5 to insert a node before a specific value. The next line contains two integers: value (existing node value) and data (value to insert).
- For choice 6 to insert a node after a specific value. The next line contains two integers: value (existing node value) and data (value to insert).
- For choice 7 to delete a node from the beginning.
- For choice 8 to delete a node from the end.
- For choice 9 to delete a node before a specific value. The next line contains an integer value representing the node before which deletion occurs.
- For choice 10 to delete a node after a specific value. The next line contains an integer value representing the node after which deletion occurs.
- For choice 11 to exit the program.

Output Format

For choice 1, print "LINKED LIST CREATED".

For choice 2, print the linked list as space-separated integers on a single line. If the list is empty, print "The list is empty".

For choice 3, 4, 5, and 6, print the updated linked list with a message indicating the insertion operation.

For choice 7, 8, 9, and 10, print the updated linked list with a message indicating the deletion operation.

For any operation that is not possible print an appropriate error message such as "Value not found in the list".

For choice 11 terminate the program.

For any invalid option, print "Invalid option! Please try again".

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 1
5
3
7
-1
2
11
Output: LINKED LIST CREATED
5 3 7

Answer

```c
// You are using GCC
#include<stdio.h>
#include<stdlib.h>
typedef struct node{
    int data;
    struct node* next;
}node;
node* cnode(int data){
    node* newn=(node*)malloc(sizeof(node));
    newn->data=data;
    newn->next=NULL;
    return newn;
}
void insertbeg(node** head,int data){
    node* newn=cnode(data);
    newn->next=*head;
    *head=newn;
}
```

```c
void insertend(node** head,int data){
    node* newn=cnode(data);
    if(!*head){
        *head=newn;
        return;
    }
    node* temp=*head;
    while(temp->next){
        temp=temp->next;
    }
    temp->next=newn;
}
void insertbef(node** head,int val,int data){
    if(!*head) return;
    if((*head)->data==val){
        insertbeg(head, data);
        return;
    }
    node *temp=*head,*prev=NULL;
    while(temp&&temp->data!=val){
        prev=temp;
        temp=temp->next;
    }
    if(!temp){
        printf("Value not found in the list\n");
        return;
    }
    prev->next=cnode(data);
    prev->next->next=temp;
}
void insertaf(node* head,int val,int data)
{
    while(head&&head->data!=val){
        head=head->next;   }
    if(!head){
        printf("Value not found in the list\n");
        return;
    }
    node* newn=cnode(data);
    newn->next=head->next;
    head->next=newn;
}
```

```c
void deletefbeg(node** head){
    if(!*head){
        return;
    }
    node* temp=*head;
    *head=(*head)->next;
    free(temp);
}
void deletefend(node** head)
{
    if(!*head)
    return;
    node* temp=*head,*prev=NULL;
    while(temp->next){
        prev=temp;
        temp=temp->next;
    }
    if(!prev)
    *head=NULL;
    else
    prev->next=NULL;
    free(temp);
}
void deletebef(node** head,int val){
    if(!*head||(*head)->data==val){
        printf("Value not found in the list\n");
        return;
    }
    node *prev2=NULL,*prev=NULL,*temp=*head;
    if((*head)->next&&(*head)->next->data==val){
        *head=(*head)->next;
        free(temp);
        return;
    }
    while(temp->next&&temp->next->data!=val){
        prev2=prev;
        prev=temp;
        temp=temp->next;
    }
    if(prev2){
        prev2->next=temp;
        free(prev);
```

```c
        }
        else{
            *head=temp;
            free(prev);
        }
    }
    void delteaf(node* head,int val){
        while(head && head->data != val)
        {
            head=head->next;
        }
        if(!head||!head->next){
            printf("Value not founf in the list\n");
            return;
        }
        node* temp=head->next;
        head->next=temp->next;
        free(temp);
    }
    void display(node* head){
        if(!head){
            printf("The list is empty\n");
            return;
        }
        while(head){
            printf(" %d",head->data);
            head=head->next;
        }
        printf("\n");
    }
    int main()
    {
        node* head=NULL;
        int choice,val,data;
        while(1){
            scanf("%d",&choice);
            switch (choice){
                case 1:
                    while(scanf("%d",&data),data!=-1)
                    insertend(&head,data);
                    printf("LINKED LIST CREATED\n");
                    break;
```

```c
        case 2:
          display(head);
          break;
        case 3:
          scanf("%d",&data);
          insertbeg(&head,data);
printf("The linked list after insertion at the beginning is:\n");
          display(head);
break;
        case 4:
          scanf("%d",&data);
          insertend(&head,data);
printf("The linked list after insertion at the end is:\n");
          display(head);
break;
        case 5:
           scanf("%d %d",&val,&data);
          insertbef(&head,val,data);
printf("The linked list after insertion before a value is:\n");
display(head);
break;
        case 6:
           scanf("%d %d",&val,&data);
          insertaf(head,val,data);
printf("The linked list after insertion after a value is:\n");
          display(head);
break;
        case 7:
          deletefbeg(&head);
printf("The linked list after deletion from the beginning is:\n");
display(head);
    break;
        case 8:
           deletefend(&head);
printf("The linked list after deletion from the end is:\n");
display(head);
    break;
        case 9:
          scanf("%d",&val);
          deletebef(&head,val);
printf("The linked list after deletion before a value is:\n");
display(head);
```

```
            break;
        case 10:
            scanf("%d",&val);
            delteaf(head,val);
    printf("The linked list after deletion after a value is:\n");
    display(head);
        break;
        case 11:
            return 0;
        default:
            printf("Invalid option! Please try again\n");
    }
  }
}
```

Status : Correct   *Marks : 1/1*

4. Problem Statement

Bharath is very good at numbers. As he is piled up with many works, he decides to develop programs for a few concepts to simplify his work. As a first step, he tries to arrange even and odd numbers using a linked list. He stores his values in a singly-linked list.

Now he has to write a program such that all the even numbers appear before the odd numbers. Finally, the list is printed in such a way that all even numbers come before odd numbers. Additionally, the even numbers should be in reverse order, while the odd numbers should maintain their original order.

Example

Input:

6

3 1 0 4 30 12

Output:

12 30 4 0 3 1

Explanation:

Even elements: 0 4 30 12

Reversed Even elements: 12 30 4 0

Odd elements: 3 1

So the final list becomes: 12 30 4 0 3 1

Input Format

The first line consists of an integer n representing the size of the linked list.

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

Output Format

The output prints the rearranged list separated by a space.

The list is printed in such a way that all even numbers come before odd numbers and the even numbers should be in reverse order, while the odd numbers should maintain their original order.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 6
3 1 0 4 30 12
Output: 12 30 4 0 3 1

Answer

```
// You are using GCC
#include<stdio.h>
#include<stdlib.h>
struct node{
    int data;
    struct node *next;
};struct node*ptr,*newnode,*head=NULL,*last=NULL;
int main()
```

```c
{int a;

    scanf("%d",&a);
    int even[a],odd[a],n=0,m=0;
    for(int i=0;i<a;i++)
    {newnode=(struct node*)malloc(sizeof(struct node));
        scanf("%d",&newnode->data);
        newnode->next=NULL;
        if(head==NULL)
        {head=newnode;}
        else
        {last->next=newnode;}
        last=newnode;
    }ptr=head;
    while(ptr!=NULL)
    {if((ptr->data) %2==0)
    {even[n++]=ptr->data;}
    else
    {odd[m++]=ptr->data;}
        ptr=ptr->next;
    }

    for(int i=n-1;i>=0;i--)
    {printf("%d ",even[i]);}
    for(int i=0;i<m;i++)
    {printf("%d ",odd[i]);}
    return 0;
}
```

Status : Correct   *Marks : 1/1*

5. Problem Statement

Write a program to manage a singly linked list. The program should allow
users to perform various operations on the linked list, such as inserting
elements at the beginning or end, deleting elements from the beginning or
end, inserting before or after a specific value, and deleting elements before
or after a specific value. After each operation, the updated linked list
should be displayed.

Input Format

The first line contains an integer choice, representing the operation to perform:

- For choice 1 to create the linked list. The next lines contain space-separated integers, with -1 indicating the end of input.
- For choice 2 to display the linked list.
- For choice 3 to insert a node at the beginning. The next line contains an integer data representing the value to insert.
- For choice 4 to insert a node at the end. The next line contains an integer data representing the value to insert.
- For choice 5 to insert a node before a specific value. The next line contains two integers: value (existing node value) and data (value to insert).
- For choice 6 to insert a node after a specific value. The next line contains two integers: value (existing node value) and data (value to insert).
- For choice 7 to delete a node from the beginning.
- For choice 8 to delete a node from the end.
- For choice 9 to delete a node before a specific value. The next line contains an integer value representing the node before which deletion occurs.
- For choice 10 to delete a node after a specific value. The next line contains an integer value representing the node after which deletion occurs.
- For choice 11 to exit the program.

Output Format

For choice 1, print "LINKED LIST CREATED".

For choice 2, print the linked list as space-separated integers on a single line. If the list is empty, print "The list is empty".

For choice 3, 4, 5, and 6, print the updated linked list with a message indicating the insertion operation.

For choice 7, 8, 9, and 10, print the updated linked list with a message indicating the deletion operation.

For any operation that is not possible print an appropriate error message such as "Value not found in the list".

For choice 11 terminate the program.

For any invalid option, print "Invalid option! Please try again".

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 1
5
3
7
-1
2
11
Output: LINKED LIST CREATED
5 3 7

Answer

```c
// You are using GCC
#include<stdio.h>
#include<stdlib.h>
 typedef struct node {
   int data;
   struct node* next;
}node;
node* cnode(int data){
   node* newn=(node*)malloc(sizeof(node));
   newn->data=data;
   newn->next=NULL;
   return newn;
}
void insertbeg(node** head,int data){
   node* newn=cnode(data);
   newn->next=*head;
   *head=newn;
}
void insertend(node** head,int data){
   node* newn=cnode(data);
   if(!*head){
      *head=newn;
      return;
   }
   node* temp=*head;
```

```c
    while(temp->next){
        temp=temp->next;
    }
    temp->next=newn;
}
void insertbef(node** head,int val,int data){
    if(!*head) return;
    if((*head)->data==val){
        insertbeg(head, data);
        return;
    }
    node *temp=*head,*prev=NULL;
    while(temp&&temp->data!=val){
        prev=temp;
        temp=temp->next;
    }
    if(!temp){
        printf("Value not found in the list\n");
        return;
    }
    prev->next=cnode(data);
    prev->next->next=temp;
}
void insertaf(node* head,int val,int data)
{
    while(head&&head->data!=val){
        head=head->next;}
    if(!head){
        printf("Value not found in the list\n");
        return;

    }
    node* newn=cnode(data);
    newn->next=head->next;
    head->next=newn;
}void deletefbeg(node** head){
    if(!*head){
        return;
    }
    node* temp=*head;
    *head=(*head)->next;
    free(temp);
```

```c
    }
    void deletefend(node** head)
    {
        if(!*head)
        return;
        node* temp=*head,*prev=NULL;
        while(temp->next){
            prev=temp;
            temp=temp->next;
        }
        if(!prev)
        *head=NULL;
        else
        prev->next=NULL;
        free(temp);
    }
    void deletebef(node** head,int val){
        if(!*head||(*head)->data==val){
            printf("Value not found in the list\n");
            return;
        }
        node *prev2=NULL,*prev=NULL,*temp=*head;
        if((*head)->next&&(*head)->next->data==val){
            *head=(*head)->next;
            free(temp);
            return;
        }
        while(temp->next&&temp->next->data!=val){
            prev2=prev;
            prev=temp;
            temp=temp->next;
        }
        if(prev2){
            prev2->next=temp;
            free(prev);
        }
        else{
            *head=temp;
            free(prev);
        }
    }
    void deleteaf(node* head,int val){
```

```c
        while(head && head->data != val)
        {
            head=head->next;
        }
        if(!head||!head->next){
            printf("Value not found in the list\n");
            return;
        }
        node* temp=head->next;
        head->next=temp->next;
        free(temp);
    }
    void display(node* head){
        if(!head){
            printf("The list is empty\n");
            return;
        }
        while(head){
            printf(" %d",head->data);
            head=head->next;
        }
        printf("\n");
    }
    int main()
    {
        node* head=NULL;
        int choice,val,data;
        while(1){
            scanf("%d",&choice);
            switch (choice){
                case 1:
                while(scanf("%d",&data),data!=-1)
                insertend(&head,data);
                printf("LINKED LIST CREATED\n");
                break;
                case 2:
                display(head);
                break;
                case 3:
                scanf("%d",&data);
                insertbeg(&head,data);
                printf("The linked list after insertion at the beginning is:\n");
```

```c
        display(head);
        break;
        case 4:
        scanf("%d",&data);
        insertend(&head,data);
        printf("The linked list after insertion at the end is:\n");
        display(head);
        break;
        case 5:
        scanf("%d %d",&val,&data);
        insertbef(&head,val,data);
        printf("The linked list after insertion before a value is:\n");
        display(head);
        break;
        case 6:
        scanf("%d %d",&val,&data);
        insertaf(head,val,data);
        printf("The linked list after insertion after a value is:\n");
        display(head);
        break;
        case 7:
        deletefbeg(&head);
        printf("The linked list after deletion from the beginning is:\n");
        display(head);
        break;
        case 8:
        deletefend(&head);
        printf("The linked list after deletion from the end is:\n");
        display(head);
        break;
        case 9:
        scanf("%d",&val);
        deletebef(&head,val);
        printf("The linked list after deletion before a value is:\n");
        display(head);
        break;
        case 10:
        scanf("%d",&val);
        deleteaf(head,val);
        printf("The linked list after deletion after a value is:\n");
        display(head);
        break;
```

```
        case 11:
        return 0;
        default:
        printf("Invalid option! Please try again\n");
    }
  }
}
```

Status : Correct   *Marks : 1/1*