Rajalakshmi Engineering College

Name: Seeralan .M
Email: 241501193@rajalakshmi.edu.in
Roll no: 241501193
Phone: 8610861705
Branch: REC
Department: I AIML AE
Batch: 2028
Degree: B.E - AI & ML

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

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

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

// Doubly Linked List Node
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;
}

// Append node to the end of the list
void append(Node** head, int data) {
    Node* newNode = createNode(data);
    if (!(*head)) {
        *head = newNode;
        return;
    }
    Node* temp = *head;
    while (temp->next != NULL)
        temp = temp->next;
    temp->next = newNode;
    newNode->prev = temp;
}

// Read input into a doubly linked list
void readInput(Node** head, int arr[], int n) {
    for (int i = 0; i < n; i++) {
        append(head, arr[i]);
    }
}

// Remove all nodes except the last occurrence of each value
void removeDuplicatesKeepLast(Node** head) {
    int seen[101] = {0};  // Assuming product IDs are from 1 to 100
    Node* temp = *head;

    // Move to the end of the list
    while (temp && temp->next != NULL) {
        temp = temp->next;
    }

    // Traverse backward, keep only the last occurrence
    while (temp) {
        if (seen[temp->data]) {
            // Remove this duplicate node
            Node* toDelete = temp;
            Node* prev = temp->prev;
            Node* next = temp->next;

            if (prev)
                prev->next = next;
            if (next)
                next->prev = prev;
            if (toDelete == *head)
                *head = next;

            temp = prev;
            free(toDelete);
        } else {
            seen[temp->data] = 1;
            temp = temp->prev;
        }
    }
}

// Print list from tail to head for reverse order output
void printListReverse(Node* head) {
    Node* temp = head;
    if (!temp)
        return;

    // Move to the tail
    while (temp->next)
        temp = temp->next;

    // Print in reverse
    while (temp) {
        printf("%d", temp->data);
        if (temp->prev)
            printf(" ");
        temp = temp->prev;
    }
}

int main() {
    int n;
    scanf("%d", &n);
    int arr[30];

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

    Node* head = NULL;
    readInput(&head, arr, n);

    removeDuplicatesKeepLast(&head);
    printListReverse(head);

    // Cleanup memory
    Node* temp = head;
    while (temp) {
        Node* next = temp->next;
        free(temp);
        temp = next;
    }

    return 0;
}
```

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

2. Problem Statement

Sam is learning about two-way linked lists. He came across a problem where he had to populate a two-way linked list and print the original as well as the reverse order of the list. Assist him with a suitable program.

Input Format

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

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

Output Format

The first line displays the message: "List in original order:"

The second line displays the elements of the doubly linked list in the original order.

The third line displays the message: "List in reverse order:"

The fourth line displays the elements of the doubly linked list in reverse order.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 5

```
1 2 3 4 5
Output: List in original order:
1 2 3 4 5
List in reverse order:
5 4 3 2 1
```

Answer

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

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

// Append node to the doubly linked list
void append(Node** head, int data) {
   Node* newNode = createNode(data);
   if (*head == NULL) {
      *head = newNode;
      return;
   }

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

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

```
// Function to print list in original order
void printOriginal(Node* head) {
    Node* temp = head;
    while (temp) {
        printf("%d", temp->data);
        if (temp->next)
            printf(" ");
        temp = temp->next;
    }
}

// Function to print list in reverse order
void printReverse(Node* head) {
    Node* temp = head;
    if (!temp)
        return;

    // Move to the last node
    while (temp->next)
        temp = temp->next;

    while (temp) {
        printf("%d", temp->data);
        if (temp->prev)
            printf(" ");
        temp = temp->prev;
    }
}

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

    Node* head = NULL;
    int value;

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

    printf("List in original order: ");
    printOriginal(head);
    printf("  ");

    printf("List in reverse order: ");
    printReverse(head);
    printf("  ");

    // Free memory
    Node* temp = head;
    while (temp) {
        Node* next = temp->next;
        free(temp);
        temp = next;
    }

    return 0;
}
```

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

3. Problem Statement

Imagine Anu is tasked with finding the middle element of a doubly linked list. Given a doubly linked list where each node contains an integer value and is inserted at the end, implement a program to find the middle element of the list. If the number of nodes is even, return the middle element pair.

Input Format

The first line of input consists of an integer N, representing the number of nodes in the doubly linked list.

The second line consists of N space-separated integers, representing the values of the nodes in the doubly linked list.

Output Format

The first line of output prints the space-separated elements of the doubly linked list.

The second line prints the middle element(s) of the doubly linked list, depending on whether the number of nodes is odd or even.

Refer to the sample outputs for the formatting specifications.

Sample Test Case

Input: 5
10 20 30 40 50

Output: 10 20 30 40 50
30

Answer

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

// Doubly Linked List Node
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 = newNode->next = NULL;
   return newNode;
}

// Function to append a node at the end of the list
void append(Node** head, int data) {
   Node* newNode = createNode(data);
   if (*head == NULL) {
      *head = newNode;
      return;
   }
   Node* temp = *head;
   while (temp->next) {
      temp = temp->next;
```

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

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

// Function to find and print the middle element(s)
void printMiddle(Node* head, int n) {
    Node* temp = head;
    int mid = n / 2;

    if (n % 2 == 1) {
        // If the number of nodes is odd, print the single middle element
        for (int i = 0; i < mid; i++) {
            temp = temp->next;
        }
        printf(" %d", temp->data);
    } else {
        // If the number of nodes is even, print the two middle elements
        for (int i = 0; i < mid - 1; i++) {
            temp = temp->next;
        }
        printf(" %d %d", temp->data, temp->next->data);
    }
}

int main() {
    int n;
    scanf("%d", &n);  // Number of nodes in the doubly linked list

    Node* head = NULL;
    int value;
    for (int i = 0; i < n; i++) {
        scanf("%d", &value);
        append(&head, value);
    }

    // Print the entire doubly linked list
    printList(head);

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

    // New line after the output
    printf("\n");

    // Free memory
    Node* temp = head;
    while (temp) {
        Node* next = temp->next;
        free(temp);
        temp = next;
    }

    return 0;
}
```

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

Rajalakshmi Engineering College

Name: Seeralan .M
Email: 241501193@rajalakshmi.edu.in
Roll no: 241501193
Phone: 8610861705
Branch: REC
Department: I AIML AE
Batch: 2028
Degree: B.E - AI & ML

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

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>

// Doubly Linked List 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 = newNode->next = NULL;
   return newNode;
}

// Function to append a new node to the doubly linked list
void append(Node** head, int data) {
   Node* newNode = createNode(data);
   if (*head == NULL) {
       *head = newNode;
       return;
   }
   Node* temp = *head;
   while (temp->next) {
       temp = temp->next;
   }
   temp->next = newNode;
   newNode->prev = temp;
}

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

// Function to check if the doubly linked list is a palindrome
int isPalindrome(Node* head) {
   // Find the tail (last node)
   Node* tail = head;
   while (tail && tail->next) {
       tail = tail->next;
   }

   // Compare nodes from both ends
   while (head && tail && head != tail && head->prev != tail) {
       if (head->data != tail->data) {
           return 0; // Not a palindrome
       }
       head = head->next;
       tail = tail->prev;
   }

   return 1; // It's a palindrome
```

```
}

int main() {
   int n;
   scanf("%d", &n);  // Read the number of nodes in the doubly linked list

   Node* head = NULL;
   int value;

   // Read the elements and append to the doubly linked list
   for (int i = 0; i < n; i++) {
      scanf("%d", &value);
      append(&head, value);
   }

   // Print the doubly linked list
   printList(head);

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

   // Free the memory used by the linked list
   Node* temp = head;
   while (temp) {
      Node* next = temp->next;
      free(temp);
      temp = next;
   }

   return 0;
}
```

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

2. 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>

// Doubly Linked List 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 = newNode->next = NULL;
   return newNode;
}

// Function to insert at the front of the doubly linked list
void insertAtFront(Node** head, int data) {
   Node* newNode = createNode(data);
   newNode->next = *head;
   if (*head != NULL) {
      (*head)->prev = newNode;
   }
   *head = newNode;
}

// Function to insert at a specific position in the doubly linked list
void insertAtPosition(Node** head, int position, int data) {
   if (position == 1) {
      insertAtFront(head, data);
      return;
   }

   Node* newNode = createNode(data);
   Node* temp = *head;
```

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

    if (temp == NULL) {
        printf("Position out of bounds.\n");
        free(newNode);
        return;
    }

    // Insert the new node after temp
    newNode->next = temp->next;
    newNode->prev = temp;

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

    temp->next = newNode;
}

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

int main() {
    int N;
    scanf("%d", &N);  // Number of initial elements

    Node* head = NULL;
    int value;

    // Read initial elements and insert at the front
    for (int i = 0; i < N; i++) {
        scanf("%d", &value);
        insertAtFront(&head, value);
    }

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

    // Read position and new data to be inserted
    int position, data;
    scanf("%d %d", &position, &data);

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

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

    // Free the memory used by the linked list
    Node* temp = head;
    while (temp) {
        Node* next = temp->next;
        free(temp);
        temp = next;
    }

    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>

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

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

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

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

    Node* temp = *head;

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

    // If node to be deleted is head
    if (temp == *head) {
        *head = temp->next;
        if (*head != NULL)
            (*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, value;
    Node* head = NULL;

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

    // Input: elements to insert at front
    for (int i = 0; i < n; i++) {
        scanf("%d", &value);
        insertAtFront(&head, value);
    }

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

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

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

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

    // Free memory
    Node* temp = head;
    while (temp) {
        Node* next = temp->next;
        free(temp);
        temp = next;
    }

    return 0;
}
```

Status : Correct **_Marks : 10/10_**

4. 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 a doubly linked list node
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 = newNode->next = NULL;
   return newNode;
}

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

// Print the list
```

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

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

    Node* tail = *head;
    while (tail->next)
        tail = tail->next;

    int splitPos = n - k;
    Node* newTail = *head;
    for (int i = 1; i < splitPos; i++) {
        newTail = newTail->next;
    }

    Node* newHead = newTail->next;

    // Update links
    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 elements
    scanf("%d", &n);

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

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

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

    // Print rotated list
    printList(head);

    // Free memory
    Node* temp;
    while (head) {
        temp = head;
        head = head->next;
        free(temp);
    }

    return 0;
}
```

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

5. 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>

// Doubly Linked List Node
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 = newNode->next = NULL;
   return newNode;
}

// Append to the end of the list
void append(Node** head, int data) {
   Node* newNode = createNode(data);
   if (*head == NULL) {
       *head = newNode;
       return;
   }
   Node* temp = *head;
   while (temp->next)
       temp = temp->next;
   temp->next = newNode;
   newNode->prev = temp;
}

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

// Print the middle element(s)
void printMiddle(Node* head, int n) {
   Node* temp = head;
   int mid = n / 2;

   // Move to mid position
   for (int i = 0; i < mid; i++) {
       temp = temp->next;
   }

   if (n % 2 == 1) {
       // Odd - print one middle
       printf("%d ", temp->data);
   } else {
       // Even - print two middles
       printf("%d %d ", temp->prev->data, temp->data);
   }
}

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

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

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

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

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

   // Free memory
   Node* temp;
   while (head) {
       temp = head;
       head = head->next;
       free(temp);
   }

   return 0;
}
```

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

Rajalakshmi Engineering College

Name: Seeralan .M
Email: 241501193@rajalakshmi.edu.in
Roll no: 241501193
Phone: 8610861705
Branch: REC
Department: I AIML AE
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

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

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

struct Node {
   char item;
   struct Node* next;
   struct Node* prev;
};

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

```
    else{
        tail->next=newnode;
        newnode->prev=tail;
        tail=newnode;
    }
   //type your code here
}
void displayForward(struct Node* head) {
    struct Node* temp=head;
    while(temp!=NULL){
        printf("%c ",temp->item);
        temp=temp->next;
    }
    printf("\n");
    //type your code here
}

void displayBackward(struct Node* tail) {
    struct Node* temp=tail;
    while(temp!=NULL){
        printf("%c ",temp->item);
        temp=temp->prev;
    }
    printf("\n");
    //type your code here
}

void freePlaylist(struct Node* head) {
    struct Node* temp=head;
    while(temp!=NULL){
        Node* nextnode=temp->next;
        free(temp);
        temp=nextnode;
    }
    head=NULL;
    tail=NULL;
    //type your code here
}

int main() {
    struct Node* playlist = NULL;
    char item;
    while (1) {
        scanf(" %c", &item);
        if (item == '-') {
            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: Seeralan .M
Email: 241501193@rajalakshmi.edu.in
Roll no: 241501193
Phone: 8610861705
Branch: REC
Department: I AIML AE
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 2_MCQ_Updated

Attempt : 1
Total Mark : 20
Marks Obtained : 19

Section 1 : MCQ

1. How do you reverse a doubly linked list?

Answer

By swapping the next and previous pointers of each node

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

2. 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***

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

Answer

2

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

4. 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***

5. 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 &lt;--&gt; 5 &lt;--&gt; 4 &lt;--&gt; 3 &lt;--&gt; 2 &lt;--&gt; 1.

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

6. 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***

7. 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***

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 first occurrence of a given data value in a doubly linked list.

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

9. 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-&gt;head = NULL; list-&gt;tail = NULL;

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

10. 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-&gt;Bwd-&gt;Fwd = X-&gt;Fwd; X-&gt;Fwd-&gt;Bwd = X-&gt;Bwd;

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

11. 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-&gt;next = head; if (head != NULL) { head-&gt;prev = newNode; } head = newNode; }

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

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

Answer

Each node has only one pointer to traverse the list back and forth

Status : Wrong ***Marks : 0/1***

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

Answer

prev

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

14. 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***

15. 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***

16. 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***

17. 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***

18. 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***

19. 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***

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

```
#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

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