Rajalakshmi Engineering College

Name: mahati sathish
Email: 240701302@rajalakshmi.edu.in
Roll no: 240701302
Phone: 9176485888
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_CY

Attempt : 1
Total Mark : 30
Marks Obtained : 20

Section 1 : Coding

1. Problem Statement

You are required to implement a program that deals with a doubly linked list.

The program should allow users to perform the following operations:

Insertion at the End: Insert a node with a given integer data at the end of the doubly linked list.Insertion at a given Position: Insert a node with a given integer data at a specified position within the doubly linked list.Display the List: Display the elements of the doubly linked list.

Input Format

The first line of input consists of an integer n, representing the number of elements to be initially inserted into the doubly linked list.

The second line consists of n space-separated integers, denoting the elements to be inserted at the end.

The third line consists of integer m, representing the new element to be inserted.

The fourth line consists of an integer p, representing the position at which the new element should be inserted (1-based indexing).

Output Format

If p is valid, display the elements of the doubly linked list after performing the insertion at the specified position.

If p is invalid, display "Invalid position" in the first line and the second line prints the original list.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
10 25 34 48 57
35
4
Output: 10 25 34 35 48 57

Answer

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

// Definition of the Node structure
struct Node

{


   int data;
   struct Node* next;
   struct Node* prev;
```

```
};

// Function to create a new node with the given data
struct Node* createNode(int data)

{


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

}

// Function to insert a node at the end of the list
void insertAtEnd(struct Node** head, int data)

{


   struct Node* newNode = createNode(data);
   if (*head == NULL)

{


      *head = newNode;


} else

{


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

{
```

```
            temp = temp->next;

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


    }

    }

    // Function to insert a node at a given position (1-based index)
    void insertAtPosition(struct Node** head, int data, int position)

    {


      struct Node* newNode = createNode(data);

      if (position < 1)

    {


        // Invalid position
        return;


    }

      if (position == 1)

    {


        // Insert at the beginning
        newNode->next = *head;
        if (*head != NULL)

    {
            (*head)->prev = newNode;

    }
        *head = newNode;


    } else

    {


        struct Node* temp = *head;
        int currentPos = 1;

        // Traverse to the position just before the desired position
        while (temp != NULL && currentPos < position - 1)

    {


            temp = temp->next;
            currentPos++;


    }

        // Check if the position is valid
        if (temp == NULL || currentPos != position - 1)

    {


            // Invalid position
            return;


    }

        // Insert the new node at the given position
        newNode->next = temp->next;
        if (temp->next != NULL)
    {

            temp->next->prev = newNode;

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

    }
    }
    // Function to display the list
    void displayList(struct Node* head)
    {

      struct Node* temp = head;
      while (temp != NULL)
    {

        printf("%d ", temp->data);
        temp = temp->next;

    }
      printf("\n");
    }
    int main()
    {
```

```
    int n, m, p;

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

    struct Node* head = NULL;

    // Insert the n initial elements at the end of the list
    for (int i = 0; i < n; i++)

  {


        int data;
        scanf("%d", &data);
        insertAtEnd(&head, data);


  }

    // Read the new element to insert and the position
    scanf("%d", &m);
    scanf("%d", &p);

    // Try inserting the new element at the given position
    if (p < 1 || p > n + 1)

  {


        // Invalid position
        printf("Invalid position\n");
        displayList(head);


  } else

  {


        // Insert at the specified position
        insertAtPosition(&head, m, p);
        displayList(head);


}

    return 0;

}
```

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

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

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


struct Node

{


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

};


struct Node* createNode(int data)

{


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

}


void insertAtEnd(struct Node** head, int data)

{


   struct Node* newNode = createNode(data);
   if (*head == NULL)

{
    *head = newNode;

} else

{


    struct Node* temp = *head;
    // Traverse to the end of the list
    while (temp->next != NULL)

{


        temp = temp->next;


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


}

}


void displayList(struct Node* head)

{


   if (head == NULL)

{


    printf("List is empty\n");
```

```
} else

{


        struct Node* temp = head;
        while (temp != NULL)

{


            printf("%d", temp->data);
            if (temp->next != NULL)

{


                printf(" ");


}
            temp = temp->next;


}
        printf("\n");


}

}

int main()

{


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

    struct Node* head = NULL;
    if (n == 0)
{

        printf("List is empty\n");

} else
{

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

            int data;
            scanf("%d", &data);
            insertAtEnd(&head, data);

}

        displayList(head);

}
    return 0;
}
```

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

3. Problem Statement

Vanessa is learning about the doubly linked list data structure and is eager to play around with it. She decides to find out how the elements are inserted at the beginning and end of the list.

Help her implement a program for the same.

Input Format

The first line of input contains an integer N, representing the size of the doubly linked list.

The next line contains N space-separated integers, each representing the values to be inserted into the doubly linked list.

Output Format

The first line of output prints the integers, after inserting them at the beginning, separated by space.

The second line prints the integers, after inserting at the end, separated by space.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
1 2 3 4 5

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

Answer

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

// Definition of a Node in the doubly linked list
struct Node

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

};

// Function to create a new node with given data
struct Node* createNode(int data)

{

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

}

// Function to insert a node at the beginning of the doubly linked list
void insertAtBeginning(struct Node** head, int data)

{

    struct Node* newNode = createNode(data);
    if (*head == NULL)

{

        *head = newNode;

} else

{
      newNode->next = *head;
      (*head)->prev = newNode;
      *head = newNode;


}

}

// Function to insert a node at the end of the doubly linked list
void insertAtEnd(struct Node** head, int data)

{


   struct Node* newNode = createNode(data);
   if (*head == NULL)

{


      *head = newNode;


} else

{


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

{


         temp = temp->next;


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

```
}

}

// Function to display the doubly linked list
void displayList(struct Node* head)

{


   struct Node* temp = head;
   while (temp != NULL)

{


      printf("%d", temp->data);
      if (temp->next != NULL)

{


         printf(" ");


}
      temp = temp->next;


}
   printf("\n");

}

int main()

{


   int N;
   scanf("%d", &N);  // Read the size of the list
```

```
    struct Node* head = NULL;

    // Read and insert elements at the beginning of the list
    int elements[N];
    for (int i = 0; i < N; i++)

 {


        scanf("%d", &elements[i]);


 }

    // Inserting at the beginning
    for (int i = N - 1; i >= 0; i--)

 {


        insertAtBeginning(&head, elements[i]);


 }
    // Display the list after insertion at the beginning
    displayList(head);

    // Reset the list for insertion at the end
    head = NULL;

    // Inserting at the end
    for (int i = 0; i < N; i++)

 {


        insertAtEnd(&head, elements[i]);


 }
```

```
    // Display the list after insertion at the end
    displayList(head);

    return 0;

}
```

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