Rajalakshmi Engineering College

Scan to verify results

Name: Shreethrudhi b
Email: 240801319@rajalakshmi.edu.in
Roll no: 240801319
Phone: 8248767847
Branch: REC
Department: I ECE AF
Batch: 2028
Degree: B.E - ECE

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 2_MCQ_Updated

Attempt : 1
Total Mark : 20
Marks Obtained : 15

Section 1 : MCQ

1. 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***

2. 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***

3. 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***

4. 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 : Wrong ***Marks : 0/1***

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

Answer

Both prev and next pointers are NULL

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

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

Answer

void addFirst(int data){&nbsp;Node* newNode = new Node(data);&nbsp;newNode-&gt;prev = head;&nbsp;head = newNode;}

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

7. How do you reverse a doubly linked list?

Answer

By swapping the next and previous pointers of each node

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

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

Answer

prev

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

9. 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***

10. 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*

11. 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*

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

Answer

2

Status : Correct *Marks : 1/1*

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

Answer

Free the memory of the node

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

14. 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***

15. 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***

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

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

17. 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***

18. 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***

19. 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***

20. 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***

Rajalakshmi Engineering College

Name: Shreethrudhi b
Email: 240801319@rajalakshmi.edu.in
Roll no: 240801319
Phone: 8248767847
Branch: REC
Department: I ECE AF
Batch: 2028
Degree: B.E - ECE

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
void insertAtEnd(struct Node** head, char item) {
  //type your code here
}
void displayForward(struct Node* head) {
   //type your code here
}

void displayBackward(struct Node* tail) {
   //type your code here
}

void freePlaylist(struct Node* head) {
   //type your code here
}*/



// Function to insert a node at the end of the doubly linked list
void insertAtEnd(struct Node** head, char item) {
   // Create a new node
   struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
   newNode->item = item;
   newNode->next = NULL;  // It will be the last node, so next is NULL
   newNode->prev = NULL;  // We'll fix this later

   if (*head == NULL) {
      // If the list is empty, newNode becomes the head
      *head = newNode;
   } else {
      // Traverse to the end of the list
      struct Node* temp = *head;
      while (temp->next != NULL) {
         temp = temp->next;
      }

      // Attach the new node to the end of the list
      temp->next = newNode;
      newNode->prev = temp; // Set the previous pointer of the new node
   }
}

// Function to display the playlist from head to tail
void displayForward(struct Node* head) {
   struct Node* temp = head;
   while (temp != NULL) {
      printf("%c ", temp->item);
      temp = temp->next;
   }
   printf("\n");
}
```

```
// Function to display the playlist from tail to head
void displayBackward(struct Node* tail) {
    struct Node* temp = tail;
    while (temp != NULL) {
        printf("%c ", temp->item);
        temp = temp->prev;
    }
    printf("\n");
}

// Function to free the memory allocated for the playlist
void freePlaylist(struct Node* head) {
    struct Node* temp;
    while (head != NULL) {
        temp = head;
        head = head->next;
        free(temp);
    }
}

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: Shreethrudhi b
Email: 240801319@rajalakshmi.edu.in
Roll no: 240801319
Phone: 8248767847
Branch: REC
Department: I ECE AF
Batch: 2028
Degree: B.E - ECE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 2_COD_Question 2

Attempt : 1
Total Mark : 10
Marks Obtained : 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

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

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

// Function to create a new node with the given participant ID
Node* createNode(int id) {
   Node* newNode = (Node*)malloc(sizeof(Node));
   newNode->id = id;
   newNode->prev = NULL;
   newNode->next = NULL;
   return newNode;
}

// Function to append a new node with the given ID to the doubly linked list
void append(Node** head, int id) {
   Node* newNode = createNode(id);

   if (*head == NULL) {
      *head = newNode;
```

```
    } else {
        Node* temp = *head;
        while (temp->next != NULL) {
            temp = temp->next;
        }
        temp->next = newNode;
        newNode->prev = temp;
    }
}

// Function to find and return the maximum ID from the list
int findMax(Node* head) {
    if (head == NULL) {
        return -1;  // Indicating empty list
    }

    int max = head->id;
    Node* temp = head;

    while (temp != NULL) {
        if (temp->id > max) {
            max = temp->id;
        }
        temp = temp->next;
    }

    return max;
}

// Function to free the memory used by the list
void freeList(Node* head) {
    Node* temp;
    while (head != NULL) {
        temp = head;
        head = head->next;
        free(temp);
    }
}

int main() {
    int n;
    scanf("%d", &n);
    if (n == 0) {
        printf("Empty list!\n");
        return 0;
    }

    Node* head = NULL;
    int id;

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

    int maxID = findMax(head);

    if (maxID == -1) {
        printf("Empty list!\n");
    } else {
        printf("%d\n", maxID);
    }

    freeList(head);  // Free the memory allocated for the list

    return 0;
}
```

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

Rajalakshmi Engineering College

Name: Shreethrudhi b
Email: 240801319@rajalakshmi.edu.in
Roll no: 240801319
Phone: 8248767847
Branch: REC
Department: I ECE AF
Batch: 2028
Degree: B.E - ECE

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

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

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

struct node* start = NULL;



// Function to traverse and print the list
void traverse() {
   struct node* temp = start;
   printf("Node Inserted\n");
   while (temp != NULL) {
      printf("%d ", temp->info);
      temp = temp->next;
```

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

// Function to insert a node at the front
void insertAtFront(int data) {
    struct node* newNode = (struct node*)malloc(sizeof(struct node));
    newNode->info = data;
    newNode->prev = NULL;
    newNode->next = start;
    if (start != NULL)
        start->prev = newNode;
    start = newNode;
}

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: Shreethrudhi b
Email: 240801319@rajalakshmi.edu.in
Roll no: 240801319
Phone: 8248767847
Branch: REC
Department: I ECE AF
Batch: 2028
Degree: B.E - ECE

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

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

// Define the structure of a doubly linked list node
struct Node {
    int data;
    struct Node* next;
    struct Node* prev;
};

// Function to insert a node at the end of the doubly linked list
void insertAtEnd(struct Node** head, int studentID) {
    // Create a new node
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = studentID;
    newNode->next = NULL;  // This will be the last node, so next is NULL
    newNode->prev = NULL;  // Will be updated later

    if (*head == NULL) {
        // If the list is empty, the new node becomes the head
        *head = newNode;
    } else {
        // Otherwise, traverse to the last node
        struct Node* temp = *head;
        while (temp->next != NULL) {
            temp = temp->next;
        }

        // Insert the new node at the end of the list
        temp->next = newNode;
        newNode->prev = temp;  // Update the previous pointer of the new node
    }
}

// Function to traverse and 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;
    scanf("%d", &N);  // Read the number of student IDs

    struct Node* head = NULL;  // Initialize an empty list

    // Read the student IDs and insert them into the doubly linked list
    for (int i = 0; i < N; i++) {
        int studentID;
        scanf("%d", &studentID);
        insertAtEnd(&head, studentID);
    }

    // Display the student IDs in the order they were inserted
    displayList(head);

    return 0;
}
```

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

Rajalakshmi Engineering College

Name: Shreethrudhi b
Email: 240801319@rajalakshmi.edu.in
Roll no: 240801319
Phone: 8248767847
Branch: REC
Department: I ECE AF
Batch: 2028
Degree: B.E - ECE

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>
```

```
// Define the structure of a doubly linked list node
struct Node {
   int data;
   struct Node* next;
   struct Node* prev;
};

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

   // If the list is empty, the new node becomes the head
   if (*head == NULL) {
      *head = newNode;
   } else {
      // Traverse to the last node
      struct Node* temp = *head;
      while (temp->next != NULL) {
         temp = temp->next;
      }

      // Insert the new node at the end
      temp->next = newNode;
      newNode->prev = temp;
   }
}

// Function to delete a node at a specific position
void deleteAtPosition(struct Node** head, int position) {
   if (*head == NULL) {
      printf("Invalid position. Try again.\n");
      return;
   }

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

   // Traverse the list to find the node at the given position
```

```
    while (temp != NULL && count < position) {
        temp = temp->next;
        count++;
    }

    // If the position is invalid (position not found)
    if (temp == NULL || count != position) {
        printf("Invalid position. Try again.\n");
        return;
    }

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

    free(temp);
}

// Function to display the list
void displayList(struct Node* head) {
    struct Node* temp = head;
    int nodeCount = 1;
    while (temp != NULL) {
        printf("node %d : %d  ", nodeCount, temp->data);
        temp = temp->next;
        nodeCount++;
    }
    printf("\n");
}
```

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

    struct Node* inventory = NULL;

    // Read n student IDs and insert them into the doubly linked list
    for (int i = 0; i < n; i++) {
        int item;
        scanf("%d", &item);
        insertAtEnd(&inventory, item);
    }

    // Display the inventory before deletion
    printf("Data entered in the list: ");
    displayList(inventory);

    // Read the position of the item to be deleted
    scanf("%d", &p);

    // Try to delete the item at position p
    deleteAtPosition(&inventory, p);

    // If deletion was successful, display the updated list
    if (p >= 1 && p <= n) {
        printf("After deletion the new list: ");
        displayList(inventory);
    }

    return 0;
}
```

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