Rajalakshmi Engineering College

Name: sakthivel M
Email: 241901095@rajalakshmi.edu.in
Roll no: 241901095
Phone: 9381167405
Branch: REC
Department: I CSE (CS) FB
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 1_MCQ

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : MCQ

1. Given the linked list: 5 -> 10 -> 15 -> 20 -> 25 -> NULL. What will be the output of traversing the list and printing each node's data?

Answer

5 10 15 20 25

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

2. The following function takes a singly linked list of integers as a parameter and rearranges the elements of the lists.

The function is called with the list containing the integers 1, 2, 3, 4, 5, 6, 7 in the given order. What will be the contents of the list after the function completes execution?

```
struct node {
   int value;
   struct node* next;
};

void rearrange (struct node* list) {
   struct node *p,q;
   int temp;
   if (! List || ! list->next) return;
   p=list; q=list->next;
   while(q) {
      temp=p->value; p->value=q->value;
      q->value=temp;p=q->next;
      q=p?p->next:0;
   }
}
```

Answer

2, 1, 4, 3, 6, 5, 7

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

3. Linked lists are not suitable for the implementation of?

Answer

Binary search

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

4. The following function reverse() is supposed to reverse a singly linked list. There is one line missing at the end of the function.

What should be added in place of "/*ADD A STATEMENT HERE*/", so that the function correctly reverses a linked list?

```
struct node {
   int data;
   struct node* next;
```

```
};
static void reverse(struct node** head_ref) {
   struct node* prev   = NULL;
   struct node* current = *head_ref;
   struct node* next;
   while (current != NULL) {
      next  = current->next;
      current->next = prev;
      prev = current;
      current = next;
   }
   /*ADD A STATEMENT HERE*/
}
```

Answer

*head_ref = prev;

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

5. Which of the following statements is used to create a new node in a singly linked list?

```
struct node {
   int data;
   struct node * next;
}
typedef struct node NODE;
NODE *ptr;
```

Answer

ptr = (NODE*)malloc(sizeof(NODE));

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

6. In a singly linked list, what is the role of the "tail" node?

Answer

It stores the last element of the list

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

7. Consider an implementation of an unsorted singly linked list. Suppose it has its representation with a head pointer only. Given the representation, which of the following operations can be implemented in O(1) time?

i) Insertion at the front of the linked list

ii) Insertion at the end of the linked list

iii) Deletion of the front node of the linked list

iv) Deletion of the last node of the linked list

Answer

I and III

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

8. Consider the singly linked list: 15 -> 16 -> 6 -> 7 -> 17. You need to delete all nodes from the list which are prime.

What will be the final linked list after the deletion?

Answer

15 -> 16 -> 6

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

9. Given a pointer to a node X in a singly linked list. If only one point is given and a pointer to the head node is not given, can we delete node X from the given linked list?

Answer

Possible if X is not last node.

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

10. Consider the singly linked list: 13 -> 4 -> 16 -> 9 -> 22 -> 45 -> 5 -> 16 -> 6, and an integer K = 10, you need to delete all nodes from the list that are less than the given integer K.

What will be the final linked list after the deletion?

Answer

13 -> 16 -> 22 -> 45 -> 16

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

Rajalakshmi Engineering College

Name: sakthivel M
Email: 241901095@rajalakshmi.edu.in
Roll no: 241901095
Phone: 9381167405
Branch: REC
Department: I CSE (CS) FB
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 : 1
Total Mark : 5
Marks Obtained : 5

Section 1 : Coding

1. 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>
#include <stdlib.h>

// Define the structure for a node in the linked list
struct Node {
    int order_id;       // The order ID
    struct Node* next;  // Pointer to the next node
};

// Function to create a new node with a given order ID
struct Node* createNode(int order_id) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->order_id = order_id;
    newNode->next = NULL;
```

```
    return newNode;
}

// Function to append a node to the linked list
void appendNode(struct Node** head, int order_id) {
    struct Node* newNode = createNode(order_id);

    // 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;
        }
        // Append the new node to the last node
        temp->next = newNode;
    }
}

// Function to print the linked list
void printList(struct Node* head) {
    struct Node* temp = head;
    while (temp != NULL) {
        printf("%d ", temp->order_id);
        temp = temp->next;
    }
    printf("\n");
}

// Function to merge two linked lists: morning and evening
struct Node* mergeLists(struct Node* morning, struct Node* evening) {
    if (morning == NULL) return evening; // If morning list is empty, return evening
list
    if (evening == NULL) return morning; // If evening list is empty, return morning
list

    // Traverse to the last node of the morning list
    struct Node* temp = morning;
    while (temp->next != NULL) {
        temp = temp->next;
```

```
    }

    // Append the evening list to the morning list
    temp->next = evening;

    return morning;
}

int main() {
    int n, m;

    // Input the number of orders in the morning list
    scanf("%d", &n);

    // Declare the morning list
    struct Node* morning = NULL;

    // Input the morning orders and append to the morning list
    for (int i = 0; i < n; i++) {
        int order_id;
        scanf("%d", &order_id);
        appendNode(&morning, order_id);
    }

    // Input the number of orders in the evening list
    scanf("%d", &m);

    // Declare the evening list
    struct Node* evening = NULL;

    // Input the evening orders and append to the evening list
    for (int i = 0; i < m; i++) {
        int order_id;
        scanf("%d", &order_id);
        appendNode(&evening, order_id);
    }

    // Merge the morning and evening lists
    struct Node* mergedList = mergeLists(morning, evening);

    // Output the merged order list
    printList(mergedList);
```

```
    return 0;
}
```

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

2. 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>

// Define the node structure for the linked list
struct Node {
    int data;
    struct Node* next;
};

// Function to create a new node
struct Node* newNode(int data) {
    struct Node* node = (struct Node*)malloc(sizeof(struct Node));
    node->data = data;
    node->next = NULL;
    return node;
}

// Function to append a node to the list
```

```
void append(struct Node** head, int data) {
    struct Node* new_node = newNode(data);
    if (*head == NULL) {
        *head = new_node;
        return;
    }
    struct Node* temp = *head;
    while (temp->next != NULL) {
        temp = temp->next;
    }
    temp->next = new_node;
}

// Function to reverse a linked list
struct Node* reverse(struct Node* head) {
    struct Node* prev = NULL;
    struct Node* current = head;
    struct Node* next = NULL;
    while (current != NULL) {
        next = current->next;
        current->next = prev;
        prev = current;
        current = next;
    }
    return prev;
}

// Function to print the linked list
void printList(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);

    struct Node *evenList = NULL, *oddList = NULL;
```

```
    int num;

    // Input the elements and segregate even and odd numbers
    for (int i = 0; i < n; i++) {
        scanf("%d", &num);
        if (num % 2 == 0) {
            append(&evenList, num); // Append even number to evenList
        } else {
            append(&oddList, num); // Append odd number to oddList
        }
    }

    // Reverse the even list
    evenList = reverse(evenList);

    // Print the final list: first evenList (reversed) then oddList
    printList(evenList);
    printList(oddList);

    return 0;
}
```

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

3. 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

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

// Define the structure for a node in the linked list
struct Node {
    int coef;   // Coefficient of the term
    int exp;    // Exponent of the term
    struct Node* next; // Pointer to the next node
};

// Function to create a new node
struct Node* createNode(int coef, int exp) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->coef = coef;
    newNode->exp = exp;
    newNode->next = NULL;
    return newNode;
}

// Function to evaluate the polynomial at a given value of x
int evaluatePolynomial(struct Node* head, int x) {
    int result = 0;
    struct Node* current = head;

    // Traverse the linked list and evaluate each term
    while (current != NULL) {
        result += current->coef * (int)pow(x, current->exp); // coef * x^exp
        current = current->next;
    }

    return result;
}

int main() {
```

```
    int degree;

    // Input the degree of the polynomial
    scanf("%d", &degree);

    // Create the linked list for the polynomial
    struct Node* head = NULL;
    struct Node* temp = NULL;

    // Input the coefficients and exponents for the polynomial terms
    for (int i = degree; i >= 0; i--) {
        int coef;
        scanf("%d", &coef);

        // Create a new node with the coefficient and exponent
        struct Node* newNode = createNode(coef, i);

        // If it's the first term, set it as the head
        if (head == NULL) {
            head = newNode;
        } else {
            temp->next = newNode;  // Add the new node to the end of the list
        }

        // Move the temp pointer to the new node
        temp = newNode;
    }

    // Input the value of x to evaluate the polynomial
    int x;
    scanf("%d", &x);

    // Evaluate the polynomial at the given x
    int result = evaluatePolynomial(head, x);

    // Output the result
    printf("%d\n", result);

    return 0;
}
```

Status : Correct *Marks : 1/1*

4. 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

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

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

struct Node* head = NULL;

void createLinkedList() {
    int data;
```

```
    while (1) {
        scanf("%d", &data);
        if (data == -1) break;

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

        if (head == NULL) {
            head = newNode;
        } else {
            struct Node* temp = head;
            while (temp->next != NULL) {
                temp = temp->next;
            }
            temp->next = newNode;
        }
    }
    printf("LINKED LIST CREATED\n");
}

void displayLinkedList() {
    if (head == NULL) {
        printf("The list is empty\n");
        return;
    }

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

void insertAtBeginning(int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = head;
    head = newNode;
    printf("The linked list after insertion at the beginning is:\n");
    displayLinkedList();
```

```
}

void insertAtEnd(int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = NULL;

    if (head == NULL) {
        head = newNode;
    } else {
        struct Node* temp = head;
        while (temp->next != NULL) {
            temp = temp->next;
        }
        temp->next = newNode;
    }
    printf("The linked list after insertion at the end is:\n");
    displayLinkedList();
}

void insertBeforeValue(int value, int data) {
    if (head == NULL) {
        printf("Value not found in the list\n");
        printf("The linked list after insertion before a value is:\n");
        displayLinkedList();
        return;
    }

    if (head->data == value) {
        insertAtBeginning(data);
        return;
    }

    struct Node* prev = NULL;
    struct Node* curr = head;

    while (curr != NULL && curr->data != value) {
        prev = curr;
        curr = curr->next;
    }

    if (curr == NULL) {
```

```
        printf("Value not found in the list\n");
        printf("The linked list after insertion before a value is:\n");
        displayLinkedList();
        return;
    }

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

    printf("The linked list after insertion before a value is:\n");
    displayLinkedList();
}

void insertAfterValue(int value, int data) {
    if (head == NULL) {
        printf("Value not found in the list\n");
        printf("The linked list after insertion after a value is:\n");
        displayLinkedList();
        return;
    }

    struct Node* temp = head;
    while (temp != NULL && temp->data != value) {
        temp = temp->next;
    }

    if (temp == NULL) {
        printf("Value not found in the list\n");
        printf("The linked list after insertion after a value is:\n");
        displayLinkedList();
        return;
    }

    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = temp->next;
    temp->next = newNode;

    printf("The linked list after insertion after a value is:\n");
    displayLinkedList();
```

```
}

void deleteFromBeginning() {
    if (head == NULL) {
        printf("The list is empty\n");
        return;
    }

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

    printf("The linked list after deletion from the beginning is:\n");
    displayLinkedList();
}

void deleteFromEnd() {
    if (head == NULL) {
        printf("The list is empty\n");
        return;
    }

    if (head->next == NULL) {
        free(head);
        head = NULL;
    } else {
        struct Node* prev = NULL;
        struct Node* curr = head;

        while (curr->next != NULL) {
            prev = curr;
            curr = curr->next;
        }

        prev->next = NULL;
        free(curr);
    }

    printf("The linked list after deletion from the end is:\n");
    displayLinkedList();
}
```

```
void deleteBeforeValue(int value) {
    if (head == NULL || head->next == NULL) {
        printf("Deletion not possible\n");
        return;
    }

    // Special case: value is in third node (delete first node)
    if (head->next->next != NULL && head->next->next->data == value) {
        struct Node* temp = head;
        head = head->next;
        free(temp);
        printf("The linked list after deletion before a value is:\n");
        displayLinkedList();
        return;
    }

    struct Node* prevPrevPrev = NULL;  // three nodes back
    struct Node* prevPrev = NULL;     // two nodes back
    struct Node* prev = head;         // one node back
    struct Node* curr = head->next;    // current node

    while (curr != NULL && curr->data != value) {
        prevPrevPrev = prevPrev;
        prevPrev = prev;
        prev = curr;
        curr = curr->next;
    }

    if (curr == NULL) {
        printf("Value not found in the list\n");
        return;
    }

    // Now delete the node two positions before curr (which is prevPrev)
    if (prevPrev == NULL) {
        printf("Deletion not possible\n");
        return;
    }

    if (prevPrevPrev == NULL) {
        // Deleting the head node
        head = prev;
```

```
        free(prevPrev);
    } else {
        prevPrevPrev->next = prev;
        free(prevPrev);
    }

    printf("The linked list after deletion before a value is:\n");
    displayLinkedList();
}
void deleteAfterValue(int value) {
    if (head == NULL) {
        printf("The list is empty\n");
        return;
    }

    struct Node* temp = head;
    while (temp != NULL && temp->data != value) {
        temp = temp->next;
    }

    if (temp == NULL || temp->next == NULL) {
        printf("Deletion not possible\n");
        return;
    }

    struct Node* nodeToDelete = temp->next;
    temp->next = nodeToDelete->next;
    free(nodeToDelete);

    printf("The linked list after deletion after a value is:\n");
    displayLinkedList();
}

int main() {
    int choice, data, value;

    while (1) {
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                createLinkedList();
```

```
            break;
        case 2:
            displayLinkedList();
            break;
        case 3:
            scanf("%d", &data);
            insertAtBeginning(data);
            break;
        case 4:
            scanf("%d", &data);
            insertAtEnd(data);
            break;
        case 5:
            scanf("%d %d", &value, &data);
            insertBeforeValue(value, data);
            break;
        case 6:
            scanf("%d %d", &value, &data);
            insertAfterValue(value, data);
            break;
        case 7:
            deleteFromBeginning();
            break;
        case 8:
            deleteFromEnd();
            break;
        case 9:
            scanf("%d", &value);
            deleteBeforeValue(value);
            break;
        case 10:
            scanf("%d", &value);
            deleteAfterValue(value);
            break;
        case 11:
            exit(0);
        default:
            printf("Invalid option! Please try again\n");
    }
}

return 0;
```

}

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

5. 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

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

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

```
struct Node* head = NULL;

void createLinkedList() {
    int data;
    while (1) {
        scanf("%d", &data);
        if (data == -1) break;

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

        if (head == NULL) {
            head = newNode;
        } else {
            struct Node* temp = head;
            while (temp->next != NULL) {
                temp = temp->next;
            }
            temp->next = newNode;
        }
    }
    printf("LINKED LIST CREATED\n");
}

void displayLinkedList() {
    if (head == NULL) {
        printf("The list is empty\n");
        return;
    }

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

void insertAtBeginning(int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
```

```
    newNode->next = head;
    head = newNode;
    printf("The linked list after insertion at the beginning is:\n");
    displayLinkedList();
}

void insertAtEnd(int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = NULL;

    if (head == NULL) {
        head = newNode;
    } else {
        struct Node* temp = head;
        while (temp->next != NULL) {
            temp = temp->next;
        }
        temp->next = newNode;
    }
    printf("The linked list after insertion at the end is:\n");
    displayLinkedList();
}

void insertBeforeValue(int value, int data) {
    if (head == NULL) {
        printf("Value not found in the list\n");
        printf("The linked list after insertion before a value is:\n");
        displayLinkedList();
        return;
    }

    if (head->data == value) {
        insertAtBeginning(data);
        return;
    }

    struct Node* prev = NULL;
    struct Node* curr = head;

    while (curr != NULL && curr->data != value) {
        prev = curr;
```

```
        curr = curr->next;
    }

    if (curr == NULL) {
        printf("Value not found in the list\n");
        printf("The linked list after insertion before a value is:\n");
        displayLinkedList();
        return;
    }

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

    printf("The linked list after insertion before a value is:\n");
    displayLinkedList();
}

void insertAfterValue(int value, int data) {
    if (head == NULL) {
        printf("Value not found in the list\n");
        printf("The linked list after insertion after a value is:\n");
        displayLinkedList();
        return;
    }

    struct Node* temp = head;
    while (temp != NULL && temp->data != value) {
        temp = temp->next;
    }

    if (temp == NULL) {
        printf("Value not found in the list\n");
        printf("The linked list after insertion after a value is:\n");
        displayLinkedList();
        return;
    }

    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = temp->next;
```

```
    temp->next = newNode;

    printf("The linked list after insertion after a value is:\n");
    displayLinkedList();
}

void deleteFromBeginning() {
    if (head == NULL) {
        printf("The list is empty\n");
        return;
    }

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

    printf("The linked list after deletion from the beginning is:\n");
    displayLinkedList();
}

void deleteFromEnd() {
    if (head == NULL) {
        printf("The list is empty\n");
        return;
    }

    if (head->next == NULL) {
        free(head);
        head = NULL;
    } else {
        struct Node* prev = NULL;
        struct Node* curr = head;

        while (curr->next != NULL) {
            prev = curr;
            curr = curr->next;
        }

        prev->next = NULL;
        free(curr);
    }
```

```
    printf("The linked list after deletion from the end is:\n");
    displayLinkedList();
}

void deleteBeforeValue(int value) {
    if (head == NULL || head->next == NULL) {
        printf("Deletion not possible\n");
        return;
    }

    if (head->next->next != NULL && head->next->next->data == value) {
        struct Node* temp = head;
        head = head->next;
        free(temp);
        printf("The linked list after deletion before a value is:\n");
        displayLinkedList();
        return;
    }

    struct Node* prevPrevPrev = NULL;
    struct Node* prevPrev = NULL;
    struct Node* prev = head;
    struct Node* curr = head->next;

    while (curr != NULL && curr->data != value) {
        prevPrevPrev = prevPrev;
        prevPrev = prev;
        prev = curr;
        curr = curr->next;
    }

    if (curr == NULL) {
        printf("Value not found in the list\n");
        return;
    }

    if (prevPrev == NULL) {
        printf("Deletion not possible\n");
        return;
    }

    if (prevPrevPrev == NULL) {
```

```
        head = prev;
        free(prevPrev);
    } else {
        prevPrevPrev->next = prev;
        free(prevPrev);
    }

    printf("The linked list after deletion before a value is:\n");
    displayLinkedList();
}

void deleteAfterValue(int value) {
    if (head == NULL) {
        printf("The list is empty\n");
        return;
    }

    struct Node* temp = head;
    while (temp != NULL && temp->data != value) {
        temp = temp->next;
    }

    if (temp == NULL || temp->next == NULL) {
        printf("Deletion not possible\n");
        return;
    }

    struct Node* nodeToDelete = temp->next;
    temp->next = nodeToDelete->next;
    free(nodeToDelete);

    printf("The linked list after deletion after a value is:\n");
    displayLinkedList();
}

int main() {
    int choice, data, value;

    while (1) {
        scanf("%d", &choice);

        switch (choice) {
```

```
            case 1:
                createLinkedList();
                break;
            case 2:
                displayLinkedList();
                break;
            case 3:
                scanf("%d", &data);
                insertAtBeginning(data);
                break;
            case 4:
                scanf("%d", &data);
                insertAtEnd(data);
                break;
            case 5:
                scanf("%d %d", &value, &data);
                insertBeforeValue(value, data);
                break;
            case 6:
                scanf("%d %d", &value, &data);
                insertAfterValue(value, data);
                break;
            case 7:
                deleteFromBeginning();
                break;
            case 8:
                deleteFromEnd();
                break;
            case 9:
                scanf("%d", &value);
                deleteBeforeValue(value);
                break;
            case 10:
                scanf("%d", &value);
                deleteAfterValue(value);
                break;
            case 11:
                exit(0);
            default:
                printf("Invalid option! Please try again\n");
        }
    }
```

```
    return 0;
}
```

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

Rajalakshmi Engineering College

Name: sakthivel M
Email: 241901095@rajalakshmi.edu.in
Roll no: 241901095
Phone: 9381167405
Branch: REC
Department: I CSE (CS) FB
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_week 1_CY

Attempt : 1
Total Mark : 30
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Rani is studying polynomials in her class. She has learned about polynomial multiplication and is eager to try it out on her own. However, she finds the process of manually multiplying polynomials quite tedious. To make her task easier, she decides to write a program to multiply two polynomials represented as linked lists.

Help Rani by designing a program that takes two polynomials as input and outputs their product polynomial. Each polynomial is represented by a linked list of terms, where each term has a coefficient and an exponent. The terms are entered in descending order of exponents.

Input Format

The first line of input consists of an integer n, representing the number of terms

in the first polynomial.

The following n lines of input consist of two integers each: the coefficient and the exponent of the term in the first polynomial.

The next line of input consists of an integer m, representing the number of terms in the second polynomial.

The following m lines of input consist of two integers each: the coefficient and the exponent of the term in the second polynomial.

Output Format

The first line of output prints the first polynomial.

The second line of output prints the second polynomial.

The third line of output prints the resulting polynomial after multiplying the given polynomials.

The polynomials should be displayed in the format, where each term is represented as ax^b, where a is the coefficient and b is the exponent.

Refer to the sample output for the exact format.

Sample Test Case

Input: 2
2 3
3 2
2
3 2
2 1

Output: 2x^3 + 3x^2
3x^2 + 2x
6x^5 + 13x^4 + 6x^3

Answer

-

Status : Skipped ***Marks : 0/10***

2. Problem Statement

Keerthi is a tech enthusiast and is fascinated by polynomial expressions. She loves to perform various operations on polynomials.

Today, she is working on a program to multiply two polynomials and delete a specific term from the result.

Keerthi needs your help to implement this program. She wants to take the coefficients and exponents of the terms of the two polynomials as input, perform the multiplication, and then allow the user to specify an exponent for deletion from the resulting polynomial, and display the result.

Input Format

The first line of input consists of an integer n, representing the number of terms in the first polynomial.

The following n lines of input consist of two integers, each representing the coefficient and the exponent of the term in the first polynomial.

The next line consists of an integer m, representing the number of terms in the second polynomial.

The following m lines of input consist of two integers, each representing the coefficient and the exponent of the term in the second polynomial.

The last line consists of an integer, representing the exponent of the term that Keerthi wants to delete from the multiplied polynomial.

Output Format

The first line of output displays the resulting polynomial after multiplication.

The second line displays the resulting polynomial after deleting the specified term.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 3
2 2
3 1
4 0
2
1 2
2 1
2
Output: Result of the multiplication: 2x^4 + 7x^3 + 10x^2 + 8x
Result after deleting the term: 2x^4 + 7x^3 + 8x

Answer

-

Status : Skipped ***Marks : 0/10***

3. Problem Statement

Timothy wants to evaluate polynomial expressions for his mathematics homework. He needs a program that allows him to input the coefficients of a polynomial based on its degree and compute the polynomial's value for a given input of x. Implement a function that takes the degree, coefficients, and the value of x, and returns the evaluated result of the polynomial.

Example

Input:

degree of the polynomial = 2

coefficient of x2 = 13

coefficient of x1 = 12

coefficient of x0 = 11

x = 1

Output:

36

Explanation:

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

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

Calculate the value of 11x0: 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 an integer representing the degree of the polynomial.

The second line consists of an integer representing the coefficient of x2.

The third line consists of an integer representing the coefficient of x1.

The fourth line consists of an integer representing the coefficient of x0.

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

Output Format

The output is an 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

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

```
// Function to evaluate the polynomial
int evaluatePolynomial(int degree, int coeffs[], int x) {
    int result = 0;

    // Iterate through the coefficients and compute the polynomial value
    for (int i = 0; i <= degree; i++) {
        result += coeffs[i] * (int)pow(x, degree - i); // Use the formula to evaluate the
polynomial
    }

    return result;
}

int main() {
    int degree;

    // Input degree of the polynomial
    scanf("%d", &degree);

    // Declare an array to store coefficients
    int coeffs[degree + 1];

    // Input the coefficients of the polynomial starting from the highest degree
    for (int i = 0; i <= degree; i++) {
        scanf("%d", &coeffs[i]);
    }

    // Input the value of x
    int x;
    scanf("%d", &x);

    // Call the evaluatePolynomial function to get the result
    int result = evaluatePolynomial(degree, coeffs, x);

    // Output the result
    printf("%d\n", result);

    return 0;
}
```

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

Rajalakshmi Engineering College

Name: sakthivel M
Email: 241901095@rajalakshmi.edu.in
Roll no: 241901095
Phone: 9381167405
Branch: REC
Department: I CSE (CS) FB
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 1_COD_Question 7

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Dev is tasked with creating a program that efficiently finds the middle element of a linked list. The program should take user input to populate the linked list by inserting each element into the front of the list and then determining the middle element.

Assist Dev, as he needs to ensure that the middle element is accurately identified from the constructed singly linked list:

If it's an odd-length linked list, return the middle element.If it's an even-length linked list, return the second middle element of the two elements.

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 integers, representing the elements of the list.

Output Format

The first line of output displays the linked list after inserting elements at the front.

The second line displays "Middle Element: " followed by the middle element of the linked list.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
10 20 30 40 50

Output: 50 40 30 20 10
Middle Element: 30

Answer

```
#include <stdio.h>
#include <stdlib.h>
struct Node {
    int data;
    struct Node* next;
};

// You are using GCC
struct Node* push(Node* head,int value){
    Node* newnode = (struct Node*)malloc(sizeof(struct Node));
    newnode->next = head;
    newnode->data= value;
    return newnode;
}
int printMiddle(struct Node*head){
    int len=0;
    Node* temp=head;
    while(temp!=NULL){
```

```
        len++;
        temp = temp->next;
    }
    int pos=len/2;
    for(int i=0;i<pos;i++){
        head= head->next;
    }
    return head->data;
}

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

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

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

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


    int middle_element = printMiddle(head);
    printf("Middle Element: %d\n", middle_element);


    current = head;
    while (current != NULL) {
        struct Node* temp = current;
        current = current->next;
        free(temp);
    }

    return 0;
```

}

Status : Correct *Marks : 10/10*

Rajalakshmi Engineering College

Name: sakthivel M
Email: 241901095@rajalakshmi.edu.in
Roll no: 241901095
Phone: 9381167405
Branch: REC
Department: I CSE (CS) FB
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 1_COD_Question 6

Attempt : 1
Total Mark : 10
Marks Obtained : 0

Section 1 : Coding

1. Problem Statement

John is tasked with creating a program to manage student roll numbers using a singly linked list.

Write a program for John that accepts students' roll numbers, inserts them at the end of the linked list, and displays the numbers.

Input Format

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

The second line consists of N space-separated integers, representing the roll numbers of students.

Output Format

The output prints the space-separated integers singly linked list, after inserting the roll numbers of students at the end.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
23 85 47 62 31

Output: 23 85 47 62 31

Answer

-

Status : Skipped ***Marks : 0/10***

Rajalakshmi Engineering College

Name: sakthivel M
Email: 241901095@rajalakshmi.edu.in
Roll no: 241901095
Phone: 9381167405
Branch: REC
Department: I CSE (CS) FB
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 1_COD_Question 5

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Imagine you are tasked with developing a simple GPA management system using a singly linked list. The system allows users to input student GPA values, insertion should happen at the front of the linked list, delete record by position, and display the updated list of student GPAs.

Input Format

The first line of input contains an integer n, representing the number of students.

The next n lines contain a single floating-point value representing the GPA of each student.

The last line contains an integer position, indicating the position at which a student record should be deleted. Position starts from 1.

Output Format

After deleting the data in the given position, display the output in the format "GPA: " followed by the GPA value, rounded off to one decimal place.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 4
3.8
3.2
3.5
4.1
2
Output: GPA: 4.1
GPA: 3.2
GPA: 3.8

Answer

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

typedef struct gpa{
    float value;
    struct gpa*next;
}Node;

Node* newnode(float value){
    Node* newgpa = (Node*)malloc(sizeof(Node));
    newgpa->value=value;
    newgpa->next=NULL;
    return newgpa;
}
Node* insertAtstart(Node* head,float value){
    Node* newgpa= newnode(value);
    newgpa->next=head;
    return newgpa;
}
```

```
void traverse(Node* head){
    while(head != NULL){
        printf("GPA: %.1f\n",head->value);
        head=head->next;
    }
}
void deleteAtPosition(Node**head,int pos){
    pos-=1;
    Node* temp = *head;
    if(pos==0){
        *head = temp->next;
        free(temp);
        return;
    }
    while(--pos){
        temp=temp->next;

    }
    Node* temp1=temp->next;
    temp->next=temp->next->next;
    free(temp1);
}
int main(){
    int n,pos;
    float value;
    scanf("%d",&n);

    Node* head=NULL;
    for(int i=0;i<n;i++){
        scanf("%f",&value);
        head=insertAtstart(head,value);
    }
    scanf("%d",&pos);
    deleteAtPosition(&head,pos);
    traverse(head);
}
```

Status : Correct *Marks : 10/10*

Rajalakshmi Engineering College

Name: sakthivel M
Email: 241901095@rajalakshmi.edu.in
Roll no: 241901095
Phone: 9381167405
Branch: REC
Department: I CSE (CS) FB
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 1_COD_Question 5

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Imagine you are tasked with developing a simple GPA management system using a singly linked list. The system allows users to input student GPA values, insertion should happen at the front of the linked list, delete record by position, and display the updated list of student GPAs.

Input Format

The first line of input contains an integer n, representing the number of students.

The next n lines contain a single floating-point value representing the GPA of each student.

The last line contains an integer position, indicating the position at which a student record should be deleted. Position starts from 1.

Output Format

After deleting the data in the given position, display the output in the format "GPA: " followed by the GPA value, rounded off to one decimal place.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 4
3.8
3.2
3.5
4.1
2
Output: GPA: 4.1
GPA: 3.2
GPA: 3.8

Answer

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

typedef struct gpa{
    float value;
    struct gpa*next;
}Node;

Node* newnode(float value){
    Node* newgpa = (Node*)malloc(sizeof(Node));
    newgpa->value=value;
    newgpa->next=NULL;
    return newgpa;
}
Node* insertAtstart(Node* head,float value){
    Node* newgpa= newnode(value);
    newgpa->next=head;
    return newgpa;
}
```

```
void traverse(Node* head){
    while(head != NULL){
        printf("GPA: %.1f\n",head->value);
        head=head->next;
    }
}
void deleteAtPosition(Node**head,int pos){
    pos-=1;
    Node* temp = *head;
    if(pos==0){
        *head = temp->next;
        free(temp);
        return;
    }
    while(--pos){
        temp=temp->next;

    }
    Node* temp1=temp->next;
    temp->next=temp->next->next;
    free(temp1);
}
int main(){
    int n,pos;
    float value;
    scanf("%d",&n);

    Node* head=NULL;
    for(int i=0;i<n;i++){
        scanf("%f",&value);
        head=insertAtstart(head,value);
    }
    scanf("%d",&pos);
    deleteAtPosition(&head,pos);
    traverse(head);
}
```

Status : Correct *Marks : 10/10*

Rajalakshmi Engineering College

Name: sakthivel M
Email: 241901095@rajalakshmi.edu.in
Roll no: 241901095
Phone: 9381167405
Branch: REC
Department: I CSE (CS) FB
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 1_COD_Question 2

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Arun is learning about data structures and algorithms. He needs your help in solving a specific problem related to a singly linked list.

Your task is to implement a program to delete a node at a given position. If the position is valid, the program should perform the deletion; otherwise, it should display an appropriate message.

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 elements of the linked list.

The third line consists of an integer x, representing the position to delete.

Position starts from 1.

Output Format

The output prints space-separated integers, representing the updated linked list after deleting the element at the given position.

If the position is not valid, print "Invalid position. Deletion not possible."

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
8 2 3 1 7
2
Output: 8 3 1 7

Answer

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

void insert(int);
void display_List();
void deleteNode(int);

struct node {
   int data;
   struct node* next;
} *head = NULL, *tail = NULL;

// You are using GCC
void insert(int value)
{
   if(head==NULL){
      head = (struct node*)malloc(sizeof(struct node));
      head->data = value;
      head->next=NULL;
   }
   else{
      struct node* temp=head;
```

```
        while(temp->next!=NULL){
            temp=temp->next;
        }
        temp->next=(struct node*)malloc(sizeof(struct node));
        temp->next->data=value;
        temp->next->next=NULL;
    }
}
void display_List(){
    struct node*list = head;
    while(list !=NULL){
        printf("%d",list->data);
        list = list->next;
    }
}
void deleteNode(int pos){
    int size=0;
    struct node*temp=head;

    while(temp!=NULL){
        size++;
        temp=temp->next;
    }
    if(size<pos){
        printf("Invalid position. Deletion not possible.",size);
    }
    else{
        pos-=1;
        if(pos==0){
            temp=head->next;
            free(head);
            head=temp;
        }
        else{
            temp=head;
            while(--pos){
                temp=temp->next;
            }
            struct node*temp1=temp->next;
            temp->next=temp->next->next;
            free(temp1);
```

```
        }
        display_List();
    }
}

int main() {
    int num_elements, element, pos_to_delete;

    scanf("%d", &num_elements);

    for (int i = 0; i < num_elements; i++) {
        scanf("%d", &element);
        insert(element);
    }

    scanf("%d", &pos_to_delete);

    deleteNode(pos_to_delete);

    return 0;
}
```

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

Rajalakshmi Engineering College

Name: sakthivel M
Email: 241901095@rajalakshmi.edu.in
Roll no: 241901095
Phone: 9381167405
Branch: REC
Department: I CSE (CS) FB
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 1_COD_Question 1

Attempt : 1
Total Mark : 10
Marks Obtained : 0

Section 1 : Coding

1. Problem Statement

Janani is a tech enthusiast who loves working with polynomials. She wants to create a program that can add polynomial coefficients and provide the sum of their coefficients.

The polynomials will be represented as a linked list, where each node of the linked list contains a coefficient and an exponent. The polynomial is represented in the standard form with descending order of exponents.

Input Format

The first line of input consists of an integer n, representing the number of terms in the first polynomial.

The following n lines of input consist of two integers each: the coefficient and the exponent of the term in the first polynomial.

The next line of input consists of an integer m, representing the number of terms in the second polynomial.

The following m lines of input consist of two integers each: the coefficient and the exponent of the term in the second polynomial.

Output Format

The output prints the sum of the coefficients of the polynomials.

Sample Test Case

Input: 3
2 2
3 1
4 0
3
2 2
3 1
4 0

Output: 18

Answer

```
// You are using GCC
#include<stdio.h>
#include<stdlib.h>
typedef struct poly{
    int coeff;
    int expon;
    struct poly* next;
}Node;

Node* newnode(int coeff,int expon){
    Node* new_node=(Node*)malloc(sizeof(Node));
    new_node->coeff=coeff;
    new_node->expon=expon;
    new_node->next=NULL;
    return new_node;
}
void insertNode(Node** head,int coeff,int expon){
    Node* temp= *head;
    if(temp==NULL){
        *head=newnode(coeff,expon);
```

```
        return;
    }
    while(temp->next!= NULL){
        temp=temp->next;
    }
    temp->next=newnode(coeff,expon);
}
int main()
{
    int n,coeff,expon;
    scanf("%d",&n);
    Node*poly1;
    Node*poly2;
    for(int i=0;i<n;i++)
    {
        scanf("%d %d",&coeff,&expon);
        insertNode(&poly1,coeff,expon);
    }
    scanf("%d",&n);
    for(int i=0;i<n;i++)
    {
        scanf("%d %d",&coeff,&expon);
        insertNode(&poly2,coeff,expon);
    }
    int sum=0;
    while(poly2!=NULL){
        sum+=poly1->coeff;
        poly1=poly1->next;
    }
    while(poly2!=NULL){
        sum+=poly2->coeff;
        poly2=poly2->next;

    }
    printf("%d",sum);
}
```

Status : Wrong *Marks : 0/10*

Rajalakshmi Engineering College

Name: sakthivel M
Email: 241901095@rajalakshmi.edu.in
Roll no: 241901095
Phone: 9381167405
Branch: REC
Department: I CSE (CS) FB
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 1_MCQ

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : MCQ

1. Given the linked list: 5 -> 10 -> 15 -> 20 -> 25 -> NULL. What will be the output of traversing the list and printing each node's data?

Answer

5 10 15 20 25

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

2. The following function takes a singly linked list of integers as a parameter and rearranges the elements of the lists.

The function is called with the list containing the integers 1, 2, 3, 4, 5, 6, 7 in the given order. What will be the contents of the list after the function completes execution?

```
struct node {
   int value;
   struct node* next;
};

void rearrange (struct node* list) {
   struct node *p,q;
   int temp;
   if (! List || ! list->next) return;
   p=list; q=list->next;
   while(q) {
      temp=p->value; p->value=q->value;
      q->value=temp;p=q->next;
      q=p?p->next:0;
   }
}
```

Answer

2, 1, 4, 3, 6, 5, 7

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

3. Linked lists are not suitable for the implementation of?

Answer

Binary search

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

4. The following function reverse() is supposed to reverse a singly linked list. There is one line missing at the end of the function.

What should be added in place of "/*ADD A STATEMENT HERE*/", so that the function correctly reverses a linked list?

```
struct node {
   int data;
   struct node* next;
```

```
};
static void reverse(struct node** head_ref) {
   struct node* prev   = NULL;
   struct node* current = *head_ref;
   struct node* next;
   while (current != NULL) {
      next  = current->next;
      current->next = prev;
      prev = current;
      current = next;
   }
   /*ADD A STATEMENT HERE*/
}
```

Answer

*head_ref = prev;

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

5. Which of the following statements is used to create a new node in a singly linked list?

```
struct node {
   int data;
   struct node * next;
}
typedef struct node NODE;
NODE *ptr;
```

Answer

ptr = (NODE*)malloc(sizeof(NODE));

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

6. In a singly linked list, what is the role of the "tail" node?

Answer

It stores the last element of the list

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

7. Consider an implementation of an unsorted singly linked list. Suppose it has its representation with a head pointer only. Given the representation, which of the following operations can be implemented in O(1) time?

i) Insertion at the front of the linked list

ii) Insertion at the end of the linked list

iii) Deletion of the front node of the linked list

iv) Deletion of the last node of the linked list

Answer

I and III

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

8. Consider the singly linked list: 15 -> 16 -> 6 -> 7 -> 17. You need to delete all nodes from the list which are prime.

What will be the final linked list after the deletion?

Answer

15 -> 16 -> 6

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

9. Given a pointer to a node X in a singly linked list. If only one point is given and a pointer to the head node is not given, can we delete node X from the given linked list?

Answer

Possible if X is not last node.

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

10. Consider the singly linked list: 13 -> 4 -> 16 -> 9 -> 22 -> 45 -> 5 -> 16 -> 6, and an integer K = 10, you need to delete all nodes from the list that are less than the given integer K.

What will be the final linked list after the deletion?

Answer

13 -> 16 -> 22 -> 45 -> 16

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