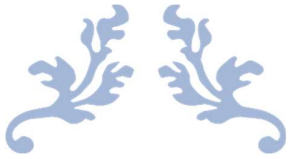

DSA ASSIGNMENT 3

PARVATHY OK

RA2311026050020

QUESTION 1: -

main.c

Share Run

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

// Structure to represent a node in the linked list
typedef struct Node {
    char name[50];
    struct Node* next;
} Node;

// Function to create a new node
Node* createNode(char* name) {
    Node* newNode = (Node*) malloc(sizeof(Node));
    strcpy(newNode->name, name);
    newNode->next = NULL;
    return newNode;
}

// Function to create the list of contacts
void createList(Node** head) {
    int numContacts;
    printf("Enter the number of contacts: ");
    scanf("%d", &numContacts);
    for (int i = 0; i < numContacts; i++) {
        char name[50];
        printf("Enter contact %d name: ", i + 1);
```

```
        char name[50];
        printf("Enter contact %d name: ", i + 1);
        scanf("%s", name);
        Node* newNode = createNode(name);
        if (*head == NULL) {
            *head = newNode;
        } else {
            Node* temp = *head;
            while (temp->next != NULL) {
                temp = temp->next;
            }
            temp->next = newNode;
        }
    }
    printf("List created successfully!\n");
}

// Function to insert a new contact
void insertContact(Node** head, int position, char* name) {
    Node* newNode = createNode(name);
    if (position == 0) {
        newNode->next = *head;
        *head = newNode;
    } else {
        Node* temp = *head;
        for (int i = 0; i < position - 1; i++) {
            temp = temp->next;
```

```
            temp = temp->next;
            if (temp == NULL) {
                printf("Invalid position. Please try again.\n");
                return;
            }
        }
        newNode->next = temp->next;
        temp->next = newNode;
    }
    printf("Contact inserted successfully!\n");
}

// Function to delete a contact by name
void deleteContactByName(Node** head, char* name) {
    Node* temp = *head;
    Node* prev = NULL;
    while (temp != NULL) {
        if (strcmp(temp->name, name) == 0) {
            if (prev == NULL) {
                *head = temp->next;
            } else {
                prev->next = temp->next;
            }
            free(temp);
            printf("Contact deleted successfully!\n");
```

```
            printf("Contact deleted successfully!\n");
            return;
        }
        prev = temp;
        temp = temp->next;
    }
    printf("Contact not found.\n");
}

// Function to delete a contact by position
void deleteContactByPosition(Node** head, int position) {
    if (position == 0 && *head != NULL) {
        Node* temp = *head;
        *head = (*head)->next;
        free(temp);
        printf("Contact deleted successfully!\n");
        return;
    }
    Node* temp = *head;
    Node* prev = NULL;
    for (int i = 0; i < position; i++) {
        prev = temp;
        temp = temp->next;
        if (temp == NULL) {
            printf("Invalid position. Please try again.\n");
            return;
```

```
            printf("Invalid position. Please try again.\n");
            return;
        }
    }
    prev->next = temp->next;
    free(temp);
    printf("Contact deleted successfully!\n");
}

// Function to traverse the list
void traverseList(Node* head) {
    printf("Current list of contacts:\n");
    int position = 1;
    while (head != NULL) {
        printf("%d. %s\n", position, head->name);
        head = head->next;
        position++;
    }
}

// Function to search for a contact
void searchContact(Node* head, char* name) {
    int position = 1;
    while (head != NULL) {
        if (strcmp(head->name, name) == 0) {
            printf("Contact found at position %d.\n", position);
```

```
            printf("Contact found at position %d.\n", position);
            return;
        }
        head = head->next;
        position++;
    }
    printf("Contact not found.\n");
}

// Function to free the list
void freeList(Node* head) {
    Node* temp;
    while (head != NULL) {
        temp = head;
        head = head->next;
        free(temp);
    }
    printf("List memory freed successfully.\n");
}

int main() {
    Node* head = NULL;
    while (1) {
        printf("\nContact Management System Menu:\n");
        printf("1. Create list\n");
        printf("2. Insert contact\n");
```

```
149         printf("2. Insert contact\n");
150         printf("3. Delete contact\n");
151         printf("4. Traverse list\n");
152         printf("5. Search contact\n");
153         printf("6. Exit\n");
154         int choice;
155         printf("Enter your choice: ");
156         scanf("%d", &choice);
157         switch (choice) {
158             case 1:
159                 createList(&head);
160                 traverseList(head);
161                 break;
162             case 2: {
163                 char name[50];
164                 int position;
165                 printf("Enter the new contact's name: ");
166                 scanf("%s", name);
167                 printf("Enter the position to insert the contact: "
                        );
168                 scanf("%d", &position);
169                 insertContact(&head, position - 1, name);
170                 traverseList(head);
171                 break;
172             }
```

```
            }
            case 3: {
                char choice;
                printf("Do you want to delete by name or by
                    position? (name/position): ");
                scanf(" %c", &choice);
                if (choice == 'n' || choice == 'N') {
                    char name[50];
                    printf("Enter the contact's name to delete: ");
                    scanf("%s", name);
                    deleteContactByName(&head, name);
                } else if (choice == 'p' || choice == 'P') {
                    int position;
                    printf("Enter the position to delete the
                        contact: ");
                    scanf("%d", &position);
                    deleteContactByPosition(&head, position - 1);
                } else {
                    printf("Invalid choice. Please try again.\n");
                }
                traverseList(head);
                break;
            }
            case 4:
                traverseList(head);
                break;
            case 5: {
                char name[50];
                printf("Enter the contact's name to search: ");
                scanf("%s", name);
                searchContact(head, name);
                break;
            }
            case 6:
                freeList(head);
                return 0;
            default:
                printf("Invalid choice. Please try again.\n");
        }
    }
    return 0;
}

```

SAMPLE OUTPUT:-

```
/tmp/dCXC3JExds.o

Contact Management System Menu:
1. Create list
2. Insert contact
3. Delete contact
4. Traverse list
5. Search contact
6. Exit
Enter your choice: 1
Enter the number of contacts: 3
Enter contact 1 name: john
Enter contact 2 name: alice
Enter contact 3 name: bob
List created successfully!
Current list of contacts:
1. john
2. alice
3. bob

Contact Management System Menu:
1. Create list
2. Insert contact
3. Delete contact
4. Traverse list
5. Search contact
```

```
5. Search contact
6. Exit
Enter your choice: 2
Enter the new contact's name: sarah
Enter the position to insert the contact: 2
Contact inserted successfully!
Current list of contacts:
1. john
2. sarah
3. alice
4. bob

Contact Management System Menu:
1. Create list
2. Insert contact
3. Delete contact
4. Traverse list
5. Search contact
6. Exit
Enter your choice: 3
Do you want to delete by name or by position? (name/position): name
Enter the contact's name to delete: Contact not found.
Current list of contacts:
1. john
2. sarah
3. alice
```

```
2. sarah
3. alice
4. bob

Contact Management System Menu:
1. Create list
2. Insert contact
3. Delete contact
4. Traverse list
5. Search contact
6. Exit
Enter your choice: 4
Current list of contacts:
1. john
2. sarah
3. alice
4. bob

Contact Management System Menu:
1. Create list
2. Insert contact
3. Delete contact
4. Traverse list
5. Search contact
6. Exit
Enter your choice: 5
```

```
Enter your choice: 5
Enter the contact's name to search: bob
Contact found at position 4.

Contact Management System Menu:
1. Create list
2. Insert contact
3. Delete contact
4. Traverse list
5. Search contact
6. Exit
Enter your choice: 6
List memory freed successfully.


=== Code Execution Successful ===
```

QUESTION 2:-

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

// Structure to represent a node in the doubly linked list
typedef struct Node {
    char name[50];
    struct Node* prev;
    struct Node* next;
} Node;

// Function to create a new node
Node* createNode(char* name) {
    Node* newNode = (Node*) malloc(sizeof(Node));
    strcpy(newNode->name, name);
    newNode->prev = NULL;
    newNode->next = NULL;
    return newNode;
}

// Function to create the list of contacts
void createList(Node** head, Node** tail) {
    int numContacts;
    printf("Enter the number of contacts: ");
    scanf("%d", &numContacts);
    for (int i = 0; i < numContacts; i++) {
```

```c
    for (int i = 0; i < numContacts; i++) {
        char name[50];
        printf("Enter contact %d name: ", i + 1);
        scanf("%s", name);
        Node* newNode = createNode(name);
        if (*head == NULL) {
            *head = *tail = newNode;
        } else {
            (*tail)->next = newNode;
            newNode->prev = *tail;
            *tail = newNode;
        }
    }
    printf("List created successfully!\n");
}

// Function to insert a new contact
void insertContact(Node** head, Node** tail, int position, char*
    name) {
    Node* newNode = createNode(name);
    if (position == 0) {
        newNode->next = *head;
        if (*head != NULL) {
            (*head)->prev = newNode;
        }
        *head = newNode;
```

```
        }
        *head = newNode;
        if (*tail == NULL) {
            *tail = newNode;
        }
    } else {
        Node* temp = *head;
        for (int i = 0; i < position - 1; i++) {
            temp = temp->next;
            if (temp == NULL) {
                printf("Invalid position. Please try again.\n");
                free(newNode);
                return;
            }
        }
        newNode->next = temp->next;
        newNode->prev = temp;
        if (temp->next != NULL) {
            temp->next->prev = newNode;
        } else {
            *tail = newNode;
        }
        temp->next = newNode;
    }
    printf("Contact inserted successfully!\n");
}
```

```
}

// Function to delete a contact by name
void deleteContactByName(Node** head, Node** tail, char* name) {
    Node* temp = *head;
    while (temp != NULL) {
        if (strcmp(temp->name, name) == 0) {
            if (temp->prev != NULL) {
                temp->prev->next = temp->next;
            } else {
                *head = temp->next;
            }
            if (temp->next != NULL) {
                temp->next->prev = temp->prev;
            } else {
                *tail = temp->prev;
            }
            free(temp);
            printf("Contact deleted successfully!\n");
            return;
        }
        temp = temp->next;
    }
    printf("Contact not found.\n");
}

```

```
99
100 // Function to delete a contact by position
101 void deleteContactByPosition(Node** head, Node** tail, int position
        ) {
102     Node* temp = *head;
103     if (position == 0 && *head != NULL) {
104         *head = (*head)->next;
105         if (*head != NULL) {
106             (*head)->prev = NULL;
107         } else {
108             *tail = NULL;
109         }
110         free(temp);
111         printf("Contact deleted successfully!\n");
112         return;
113     }
114     for (int i = 0; i < position; i++) {
115         temp = temp->next;
116         if (temp == NULL) {
117             printf("Invalid position. Please try again.\n");
118             return;
119         }
120     }
121     if (temp->prev != NULL) {
122         temp->prev->next = temp->next;
```

```
122         temp->prev->next = temp->next;
123     }
124     if (temp->next != NULL) {
125         temp->next->prev = temp->prev;
126     } else {
127         *tail = temp->prev;
128     }
129     free(temp);
130     printf("Contact deleted successfully!\n");
131 }
132 
133 // Function to traverse the list in forward direction
134 void traverseListForward(Node* head) {
135     printf("Current list of contacts (Forward Traversal):\n");
136     int position = 1;
137     while (head != NULL) {
138         printf("%d. %s\n", position, head->name);
139         head = head->next;
140         position++;
141     }
142 }
143 
144 // Function to traverse the list in reverse direction
145 void traverseListReverse(Node* tail) {
146     printf("Current list of contacts (Reverse Traversal):\n");
147     int position = 1;
```

```c
    int position = 1;
    while (tail != NULL) {
        printf("%d. %s\n", position, tail->name);
        tail = tail->prev;
        position++;
    }
}

// Function to search for a contact
void searchContact(Node* head, char* name) {
    int position = 1;
    while (head != NULL) {
        if (strcmp(head->name, name) == 0) {
            printf("Contact found at position %d.\n", position);
            return;
        }
        head = head->next;
        position++;
    }
    printf("Contact not found.\n");
}

// Function to free the list
void freeList(Node* head) {
    Node* temp;
    while (head != NULL) {
```

```c
    Node* temp;
    while (head != NULL) {
        temp = head;
        head = head->next;
        free(temp);
    }
    printf("List memory freed successfully.\n");
}

int main() {
    Node* head = NULL;
    Node* tail = NULL;
    while (1) {
        printf("\nContact Management System Menu:\n");
        printf("1. Create list\n");
        printf("2. Insert contact\n");
        printf("3. Delete contact\n");
        printf("4. Traverse list\n");
        printf("5. Search contact\n");
        printf("6. Exit\n");
        int choice;
        printf("Enter your choice: ");
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                createList(&head, &tail);
```

```
195             case 1:
196                 createList(&head, &tail);
197                 traverseListForward(head);
198                 traverseListReverse(tail);
199                 break;
200             case 2: {
201                 char name[50];
202                 int position;
203                 printf("Enter the new contact's name: ");
204                 scanf("%s", name);
205                 printf("Enter the position to insert the contact: "
                        );
206                 scanf("%d", &position);
207                 insertContact(&head, &tail, position - 1, name);
208                 traverseListForward(head);
209                 traverseListReverse(tail);
210                 break;
211             }
212             case 3: {
213                 char choice;
214                 printf("Do you want to delete by name or by
                        position? (name/position): ");
215                 scanf(" %c", &choice);
216                 if (choice == 'n' || choice == 'N') {
217                     char name[50];
218                     printf("Enter the contact's name to delete: ");
219                     scanf("%s", name);
```

```
                printf("Enter the contact's name to delete: ");
                scanf("%s", name);
                deleteContactByName(&head, &tail, name);
            } else if (choice == 'p' || choice == 'P') {
                int position;
                printf("Enter the position to delete the
                    contact: ");
                scanf("%d", &position);
                deleteContactByPosition(&head, &tail, position
                    - 1);
            } else {
                printf("Invalid choice. Please try again.\n");
            }
            traverseListForward(head);
            traverseListReverse(tail);
            break;
        }
        case 4:
            traverseListForward(head);
            traverseListReverse(tail);
            break;
        case 5: {
            char name[50];
            printf("Enter the contact's name to search: ");
            scanf("%s", name);
            searchContact(head, name);
            break;
        }
        case 6:
            freeList(head);
            return 0;
        default:
            printf("Invalid choice. Please try again.\n");
    }
}
return 0;
}
```

SAMPLE OUTPUT:-

```
/tmp/mAJHus5JsX.o

Contact Management System Menu:
1. Create list
2. Insert contact
3. Delete contact
4. Traverse list
5. Search contact
6. Exit
Enter your choice: 1
Enter the number of contacts: 3
Enter contact 1 name: john
Enter contact 2 name: alice'
Enter contact 3 name: bob
List created successfully!
Current list of contacts (Forward Traversal):
1. john
2. alice
3. bob
Current list of contacts (Reverse Traversal):
1. bob
2. alice
3. john

Contact Management System Menu:
1. Create list
```

```
Contact Management System Menu:
1. Create list
2. Insert contact
3. Delete contact
4. Traverse list
5. Search contact
6. Exit
Enter your choice: 2
Enter the new contact's name: sarah
Enter the position to insert the contact: 2
Contact inserted successfully!
Current list of contacts (Forward Traversal):
1. john
2. sarah
3. alice
4. bob
Current list of contacts (Reverse Traversal):
1. bob
2. alice
3. sarah
4. john

Contact Management System Menu:
1. Create list
2. Insert contact
3. Delete contact
4. Traverse list
```

```
3. Delete contact
4. Traverse list
5. Search contact
6. Exit
Enter your choice: 3
Do you want to delete by name or by position? (name/position): p
Enter the position to delete the contact: 1
Contact deleted successfully!
Current list of contacts (Forward Traversal):
1. sarah
2. alice
3. bob
Current list of contacts (Reverse Traversal):
1. bob
2. alice
3. sarah

Contact Management System Menu:
1. Create list
2. Insert contact
3. Delete contact
4. Traverse list
5. Search contact
6. Exit
Enter your choice: 4
Current list of contacts (Forward Traversal):
```

```
Current list of contacts (Forward Traversal):
1. sarah
2. alice
3. bob
Current list of contacts (Reverse Traversal):
1. bob
2. alice
3. sarah

Contact Management System Menu:
1. Create list
2. Insert contact
3. Delete contact
4. Traverse list
5. Search contact
6. Exit
Enter your choice: 5
Enter the contact's name to search: bob
Contact found at position 3.

Contact Management System Menu:
1. Create list
2. Insert contact
3. Delete contact
4. Traverse list
5. Search contact
5. Search contact
6. Exit
Enter your choice: 6
List memory freed successfully.


=== Code Execution Successful ===
```