Qno.1)

#include <stdio.h>

#include <stdlib.h>

struct Node {

int data;

struct Node \*next;

};

void deleteNode(struct Node \*\*head\_ref, int key) {

// Store head node

struct Node \*temp = \*head\_ref, \*prev;

// If head node contains the key to be deleted

if (temp != NULL && temp->data == key) {

\*head\_ref = temp->next; // Changed head

free(temp); // free old head

return;

}

// Search for the key to be deleted, keep track of the

// previous node as we need to change 'prev->next'

while (temp != NULL && temp->data != key) {

prev = temp;

temp = temp->next;

}

// If key was not present in linked list

if (temp == NULL) return;

// Unlink the node from linked list

prev->next = temp->next;

free(temp); // Free memory

}

void push(struct Node \*\*head\_ref, int new\_data) {

struct Node \*new\_node = (struct Node \*)malloc(sizeof(struct Node));

new\_node->data = new\_data;

new\_node->next = (\*head\_ref);

(\*head\_ref) = new\_node;

}

void printList(struct Node \*node) {

while (node != NULL) {

printf("%d ", node->data);

node = node->next;

}

printf("\n");

}

int main() {

struct Node \*head = NULL;

push(&head, 1);

push(&head, 2);

push(&head, 1);

push(&head, 3);

printf("Linked List before deletion: ");

printList(head);

deleteNode(&head, 1);

printf("Linked List after deletion: ");

printList(head);

return 0;

}

Qno.2)

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

struct Node {

int data;

struct Node \*next;

};

void deleteDuplicates(struct Node \*head) {

struct Node \*current = head;

struct Node \*next\_next;

// Check for empty list

if (current == NULL) {

return;

}

while (current->next != NULL) {

// Compare current node with next node

if (current->data == current->next->data) {

next\_next = current->next->next;

free(current->next);

current->next = next\_next;

} else {

current = current->next;

}

}

}

void push(struct Node \*\*head\_ref, int new\_data) {

struct Node \*new\_node = (struct Node \*)malloc(sizeof(struct Node));

new\_node->data = new\_data;

new\_node->next = (\*head\_ref);

(\*head\_ref) = new\_node;

}

void printList(struct Node \*node) {

while (node != NULL) {

printf("%d ", node->data);

node = node->next;

}

printf("\n");

}

int main() {

struct Node \*head = NULL;

push(&head, 2);

push(&head, 1);

push(&head, 1);

push(&head, 3);

push(&head, 2);

printf("Linked List before deletion: ");

printList(head);

deleteDuplicates(head);

printf("Linked List after deletion: ");

printList(head);

return 0;

}

Qno.3)

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

struct Node {

int data;

struct Node \*next;

};

void deleteDuplicates(struct Node \*head) {

struct Node \*current = head;

struct Node \*runner;

struct Node \*prev;

// Check for empty list

if (current == NULL) {

return;

}

while (current != NULL) {

prev = current;

runner = current->next;

while (runner != NULL) {

if (current->data == runner->data) {

// Remove the duplicate node

prev->next = runner->next;

free(runner);

runner = prev->next;

} else {

prev = runner;

runner = runner->next;

}

}

current = current->next;

}

}

void push(struct Node \*\*head\_ref, int new\_data) {

struct Node \*new\_node = (struct Node \*)malloc(sizeof(struct Node));

new\_node->data = new\_data;

new\_node->next = (\*head\_ref);

(\*head\_ref) = new\_node;

}

void printList(struct Node \*node) {

while (node != NULL) {

printf("%d ", node->data);

node = node->next;

}

printf("\n");

}

int main() {

struct Node \*head = NULL;

push(&head, 2);

push(&head, 1);

push(&head, 1);

push(&head, 3);

push(&head, 2);

printf("Linked List before deletion: ");

printList(head);

deleteDuplicates(head);

printf("Linked List after deletion: ");

printList(head);

return 0;

}

Qno.4)

#include <stdio.h>

#include <stdlib.h>

struct Node {

int data;

struct Node \*next;

};

void reverse(struct Node \*\*head\_ref) {

struct Node \*prev = NULL;

struct Node \*current = \*head\_ref;

struct Node \*next = NULL;

while (current != NULL) {

next = current->next;

current->next = prev;

prev = current;

current = next;

}

\*head\_ref = prev;

}

void push(struct Node \*\*head\_ref, int new\_data) {

struct Node \*new\_node = (struct Node \*)malloc(sizeof(struct Node));

new\_node->data = new\_data;

new\_node->next = (\*head\_ref);

(\*head\_ref) = new\_node;

}

void printList(struct Node \*node) {

while (node != NULL) {

printf("%d ", node->data);

node = node->next;

}

printf("\n");

}

int main() {

struct Node \*head = NULL;

push(&head, 1);

push(&head, 2);

push(&head, 3);

printf("Linked List before reversing: ");

printList(head);

reverse(&head);

printf("Linked List after reversing: ");

printList(head);

return 0;

}

Qno.5)

#include <stdio.h>

#include <stdlib.h>

struct Node {

int data;

struct Node \*next;

};

float average(struct Node \*head) {

float sum = 0;

int count = 0;

while (head != NULL) {

sum += head->data;

count++;

head = head->next;

}

return sum / count;

}

void push(struct Node \*\*head\_ref, int new\_data) {

struct Node \*new\_node = (struct Node \*)malloc(sizeof(struct Node));

new\_node->data = new\_data;

new\_node->next = (\*head\_ref);

(\*head\_ref) = new\_node;

}

int main() {

struct Node \*head = NULL;

push(&head, 1);

push(&head, 2);

push(&head, 3);

float avg = average(head);

printf("Average of the linked list: %f\n", avg);

return 0;

}