## Exercise 1

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

typedef struct Node {

int value;

struct Node\* next;

} Node;

Node\* createNode(int value)

{

Node\* newNode = (Node\*)malloc(sizeof(Node));

if (!newNode)

{

printf("Memory allocation failed");

return 1;

}

newNode->value = value;

newNode->next = NULL;

return newNode;

}

void printList(Node\* head)

{

Node\* current = head;

while (current != NULL)

{

printf("%d -> ", current->value);

current = current->next;

}

}

void freeList(Node\* head)

{

Node\* current = head;

while (current != NULL)

{

Node\* temp = current;

current = current->next;

free(temp);

}

}

Node\* deleteInRange(Node\* head, int mink, int maxk)

{

Node dummy;

dummy.next = head;

Node\* p = &dummy;

while (p->next != NULL && p->next->value <= mink)

{

p = p->next;

}

if (p->next == NULL)

{

return head;

}

Node\* startDelete = p->next;

Node\* q = startDelete;

while (q != NULL && q->value < maxk)

{

q = q->next;

}

p->next = q;

Node\* current = startDelete;

while (current != q) {

Node\* temp = current;

current = current->next;

printf("Deleting node with value: %d\n", temp->value);

free(temp);

}

return dummy.next;

}

## Exercise 2

Node\* reverseList(Node\* head)

{

Node\* prev = NULL;

Node\* curr = head;

Node\* next\_temp = NULL;

while (curr != NULL)

{

next\_temp = curr->next;

curr->next = prev;

prev = curr;

curr = next\_temp;

}

return prev;

}