LAB 4

Q1.

```
#include <stdio.h> #include
<stdlib.h>#include
<string.h>
struct Contacts
{
        char name[100]; char
        email[100];char
        p_num[12];
};
int main(){
        struct Contacts *cont = malloc(1*sizeof(struct Contacts)); // anarray of structs
        int i = 0, choice;
        // char del_name[100];int
        del_num;
        int j;
        while(1){
                printf("\n");
                printf("What would you like to do?\n1. Add a contact\n2. Deletea contact\n3. Print
contacts\n4. Delete Address Book\n");
                scanf("%d", &choice);
                printf("\n");
                switch (choice)
                {
                case 1:

                        printf("Enter the name of the contact: ");scanf("%s", cont[i].name);
                        printf("Enter the email of the contact: ");scanf("%s",
                        cont[i].email);
                        printf("Enter the phone number of the contact: ");scanf("%s", cont[i].p_num);
                        i++;
                        cont = realloc(cont, sizeof(struct Contacts) * (i));break;

                case 2:
                        printf("\nEnter the serial number of the contact you wouldlike to delete: ");
                        printf("\n");
                        for(int j = 0; j<i; j++){
                                printf("%d.   %s                    %s                %s\n", j+1, cont[j].name,
cont[j].email, cont[j].p_num);
                        }
                        scanf("%d", &del_num);
                        printf("\n");
                        for(j=del_num-1; j<i-1; j++){
```

```
                    cont[j] = cont[j+1];
                    }
                    i--;
                break;

            case 3:
                for(int j = 0; j<i; j++){
                    printf("%s            %s          %s\n", cont[j].name,
cont[j].email, cont[j].p_num);
                }
                break;

            case 4:
                free(cont);
                return 0;

            default:
                printf("Enter a valid option!");
            }
        }
}
```

Q2.

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

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

struct Node *createNode(int data)
{
        struct Node *newNode = malloc(sizeof(struct Node));if (newNode == NULL)
        {
                fprintf(stderr, "Memory allocation failed!");exit(0);
        }
        newNode->data    =    data;
        newNode->next = NULL;return
        newNode;
}

struct Node *insertAfter(struct Node *head, int data, int search)
{
        struct Node *current = head;
        while (current != NULL && current->data != search)
        {
                current = current->next;
        }
        if (current->data == search)
        {
                struct Node *new = createNode(data);new->next =
                current->next;
                current->next = new;
        }
        else
        {
                printf("Node not found!\n");
        }
        return head;
}

struct Node *insertAtTheEnd(struct Node *head, int data)
{
        struct Node *newNode = createNode(data);if (head == NULL)
        {
                return newNode;
        }
        struct Node *current = head; while (current-
        >next != NULL)
        {
                current = current->next;
        }
        current->next = newNode;return
        head;
}

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

int main()
{
        struct Node *head1 = createNode(1);insertAfter(head1, 2,
        1);
        insertAfter(head1, 3, 2);
        printList(head1);

        struct Node *head2 = createNode(2);insertAfter(head2, 3,
        2);
        insertAfter(head2, 4, 3);
        printList(head2);

        struct Node *temp1 = head1;struct
        Node *temp2 = head2;struct Node
        *head3 = NULL; int val;
        while (temp1 != NULL && temp2 != NULL)
```

```
    {
        if (temp1->data < temp2->data)
        {
            val = temp1->data; temp1 =
            temp1->next;
        }
        else
        {
            val = temp2->data; temp2 =
            temp2->next;
        }
        head3 = insertAtTheEnd(head3, val);
    }
    if (temp1 != NULL)
    {
        head3 = insertAtTheEnd(head3, temp1->data);temp1 = temp1->next;
    }
    if (temp2 != NULL)
    {
        head3 = insertAtTheEnd(head3, temp2->data);temp2 = temp2->next;
    }

    printList(head3);return
    0;
}
```

Q3.

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

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

struct Node *createNode(int data)
{
    struct Node *newNode = malloc(sizeof(struct Node));if (newNode == NULL)
    {
        fprintf(stderr, "Memory allocation failed!");exit(0);
    }
    newNode->data    =    data;
    newNode->next = NULL;return
    newNode;
}

struct Node *insertAfter(struct Node *head, int data, int search)
{
    struct Node *current = head;
```

```
        while (current != NULL && current->data != search)
        {
                current = current->next;
        }
        if (current->data == search)
        {
                struct Node *new = createNode(data);new->next =
                current->next;
                current->next = new;
        }
        else
        {
                printf("Node not found!\n");
        }
        return head;
}

int nodeCount(struct Node* head){struct Node*
        current = head; int count = 0;
        while(current != NULL){count++;
                current = current->next;
        }
        return count;
}

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

int main()
{
        struct Node *head = createNode(1);insertAfter(head, 2, 1);
        insertAfter(head, 3, 2);
        printList(head);
        int arr[50], i = 0, count = nodeCount(head);struct Node *temp    =
        head; while(temp!=NULL){
                arr[i] = temp->data;temp =
                temp->next; i++;
        }
        for(int i = 0; i < count; i++){printf("%d\n", arr[i]);
        }
        return 0;
```

```
}
```

Q4.

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

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

struct Node *createNode(int data)
{
        struct Node *newNode = malloc(sizeof(struct Node));if (newNode == NULL)
        {
                fprintf(stderr, "Memory allocation failed!");exit(0);
        }
        newNode->data       =       data;
        newNode->next = NULL;return
        newNode;
}
struct Node *insertAfter(struct Node *head, int data, int search)
{
        struct Node *current = head;
        while (current != NULL && current->data != search)
        {
                current = current->next;
        }
        if (current->data == search)
        {
                struct Node *new = createNode(data);new->next =
                current->next;
                current->next = new;
        }
        else
        {
                printf("Node not found!\n");
        }
        return head;
}
int nodeCount(struct Node* head){struct Node*
        current = head; int count = 0;
        while(current != NULL){count++;
                current = current->next;
        }
        return count;
}
```

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

int main()
{
        struct Node *head = createNode(1);insertAfter(head, 2, 1);
        insertAfter(head, 3, 2);
        insertAfter(head, 4, 3);
        insertAfter(head, 5, 4);
        printList(head);

        int count = nodeCount(head), i = 0;struct Node *temp =
        head;
        struct Node *temp2;
        while(temp!=NULL){
                        temp2 = temp->next;
                        if(temp2!=NULL){
                                temp->next = temp2->next;
                        }
                        free(temp2); temp =
                temp->next;i+=2;
        }
        printList(head);return
        0;
}
```