Informatics II, Spring 2023, Exercise 6

Publication of exercise: March 27, 2023
Publication of solution: April 02, 2023
Exercise classes: April 18 - April 21, 2023

Task 1

C B A D C C C

Task 2

```c
#include<stdio.h>
#include<stdlib.h>
#define N 5

void print(int *arr, int n); // (a)

int* reverse(int *arr, int n); // (b)

int* prepend(int *arr, int v); // (c)

void print(int *arr, int n) {
    for (int i = 0; i < n; i++) {
        printf("%d␣", *(arr + i));
    }
    printf("\n");
}

int* reverse(int *arr, int n) {
    int *rev;
    rev = malloc(N * sizeof(int));

    for (int i = 0; i < N; i++) {
        *(rev + i) = *(arr + n - 1 - i);
    }

    return rev;
}

int* prepend(int *arr, int v) {
    int *pre;
    pre = malloc((N + 1) * sizeof(int));
    *(pre) = v;

    for (int i = 0; i < N; i++) {
        *(pre + i + 1) = *(arr + i);
    }

    return pre;
```

```
}

int main() {
    int *arr;
    arr = malloc(N * sizeof(int));

    for (int i = 0; i < N; i++) {
        arr[i] = i;
    }

    printf("The original: ");
    print(arr, N);

    int *reversed = reverse(arr, N);

    free(arr);

    printf("The reversed: ");
    print(reversed, N);

    int *prepended = prepend(reversed, 5);

    free(reversed);

    printf("The prepended: ");
    print(prepended, N + 1);
}
```

Task 3

```
#include <stdio.h>
#include <stdlib.h>
#define N 10

struct node {
    int key;
    struct node* next;
};

struct node* convertArraytoLinkedList(int *arr, int n); // convert a given array to a
    linked list

void print(struct node* curr); // a printing function

struct node* reverseLinkedList(struct node* head); // wrapper function of the recursive

struct node* reverse(struct node* prev, struct node* curr); // the recursive

struct node* convertArraytoLinkedList(int *arr, int n) {
    struct node* head;
    head = malloc(sizeof(struct node));
    head->key = *(arr);

    struct node* curr = head;
    for (int i = 1; i < N; i++) {
        curr->next = malloc(sizeof(struct node));
        curr = curr->next;
        curr->key = arr[i];
        curr->next = NULL;
    }
    return head;
```

```
}

struct node* reverseLinkedList(struct node* head) {
    struct node** dummy = malloc(sizeof(struct node*));
    *dummy = malloc(sizeof(struct node));
    reverse(*dummy, head);

    return (*dummy)->next;
}

struct node* reverse(struct node* prev, struct node* curr) {
    // base case
    if (curr->next == NULL) {
        prev->next = curr;
        return curr;
    }

    // recursive logic
    struct node* tempTail = reverse(prev, curr->next);
    tempTail->next = curr;
    curr->next = NULL;
    return curr;
}

void print(struct node* curr) {
    while (curr != NULL) {
        printf("%d ", curr->key);
        curr = curr->next;
    }
}

int main() {
    /* Generating an array of N random integers */
    int *arr;
    arr = malloc(N * sizeof(int));
    for (int i = 0; i < N; i++) {
        *(arr + i) = rand();
    }


    struct node *head = convertArraytoLinkedList(arr, N);

    printf("\nThe original: ");
    print(head);

    struct node* newHead = reverseLinkedList(head);

    printf("\nThe reversed: ");
    print(newHead);

    return 1;
}
```