

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

```
1 #include<stdio.h>
2 #include<stdlib.h>
3 #define N 5
4
5 void print(int *arr, int n); // (a)
6
7 int* reverse(int *arr, int n); // (b)
8
9 int* prepend(int *arr, int v); // (c)
10
11 void print(int *arr, int n) {
12     for (int i = 0; i < n; i++) {
13         printf("%d\\", *(arr + i));
14     }
15     printf("\\n");
16 }
17
18 int* reverse(int *arr, int n) {
19     int *rev;
20     rev = malloc(N * sizeof(int));
21
22     for (int i = 0; i < N; i++) {
23         *(rev + i) = *(arr + n - 1 - i);
24     }
25
26     return rev;
27 }
28
29 int* prepend(int *arr, int v) {
30     int *pre;
31     pre = malloc((N + 1) * sizeof(int));
32     *(pre) = v;
33
34     for (int i = 0; i < N; i++) {
35         *(pre + i + 1) = *(arr + i);
36     }
37
38     return pre;
```

```
39 }
40
41 int main() {
42     int *arr;
43     arr = malloc(N * sizeof(int));
44
45     for (int i = 0; i < N; i++) {
46         arr[i] = i;
47     }
48
49     printf("The original:");
50     print(arr, N);
51
52     int *reversed = reverse(arr, N);
53
54     free(arr);
55
56     printf("The reversed:");
57     print(reversed, N);
58
59     int *prepending = prepend(reversed, 5);
60
61     free(reversed);
62
63     printf("The prepended:");
64     print(prepending, N + 1);
65 }
```

Task 3

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #define N 10
4
5 struct node {
6     int key;
7     struct node* next;
8 };
9
10 struct node* convertArraytoLinkedList(int *arr, int n); // convert a given array to a
    linked list
11
12 void print(struct node* curr); // a printing function
13
14 struct node* reverseLinkedList(struct node* head); // wrapper function of the recursive
15
16 struct node* reverse(struct node* prev, struct node* curr); // the recursive
17
18 struct node* convertArraytoLinkedList(int *arr, int n) {
19     struct node* head;
20     head = malloc(sizeof(struct node));
21     head->key = *(arr);
22
23     struct node* curr = head;
24     for (int i = 1; i < N; i++) {
25         curr->next = malloc(sizeof(struct node));
26         curr = curr->next;
27         curr->key = arr[i];
28         curr->next = NULL;
29     }
30     return head;
```

```
31 }
32
33 struct node* reverseLinkedList(struct node* head) {
34     struct node** dummy = malloc(sizeof(struct node*));
35     *dummy = malloc(sizeof(struct node));
36     reverse(*dummy, head);
37
38     return (*dummy)->next;
39 }
40
41 struct node* reverse(struct node* prev, struct node* curr) {
42     // base case
43     if (curr->next == NULL) {
44         prev->next = curr;
45         return curr;
46     }
47
48     // recursive logic
49     struct node* tempTail = reverse(prev, curr->next);
50     tempTail->next = curr;
51     curr->next = NULL;
52     return curr;
53 }
54
55 void print(struct node* curr) {
56     while (curr != NULL) {
57         printf("%d_", curr->key);
58         curr = curr->next;
59     }
60 }
61
62 int main() {
63     /* Generating an array of N random integers */
64     int *arr;
65     arr = malloc(N * sizeof(int));
66     for (int i = 0; i < N; i++) {
67         *(arr + i) = rand();
68     }
69
70
71     struct node *head = convertArraytoLinkedList(arr, N);
72
73     printf("\nThe original:");
74     print(head);
75
76     struct node* newHead = reverseLinkedList(head);
77
78     printf("\nThe reversed:");
79     print(newHead);
80
81     return 1;
82 }
```