Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Інститут прикладного системного аналізу
Кафедра системного проектування сервісів

ЗВІТ

про виконання комп'ютерного практикуму № 4
з дисципліни «Алгоритми і структури даних»

Виконав:
Студент І курсу
Групи ДА-72
Хоменко Є.С
Варіант № 27

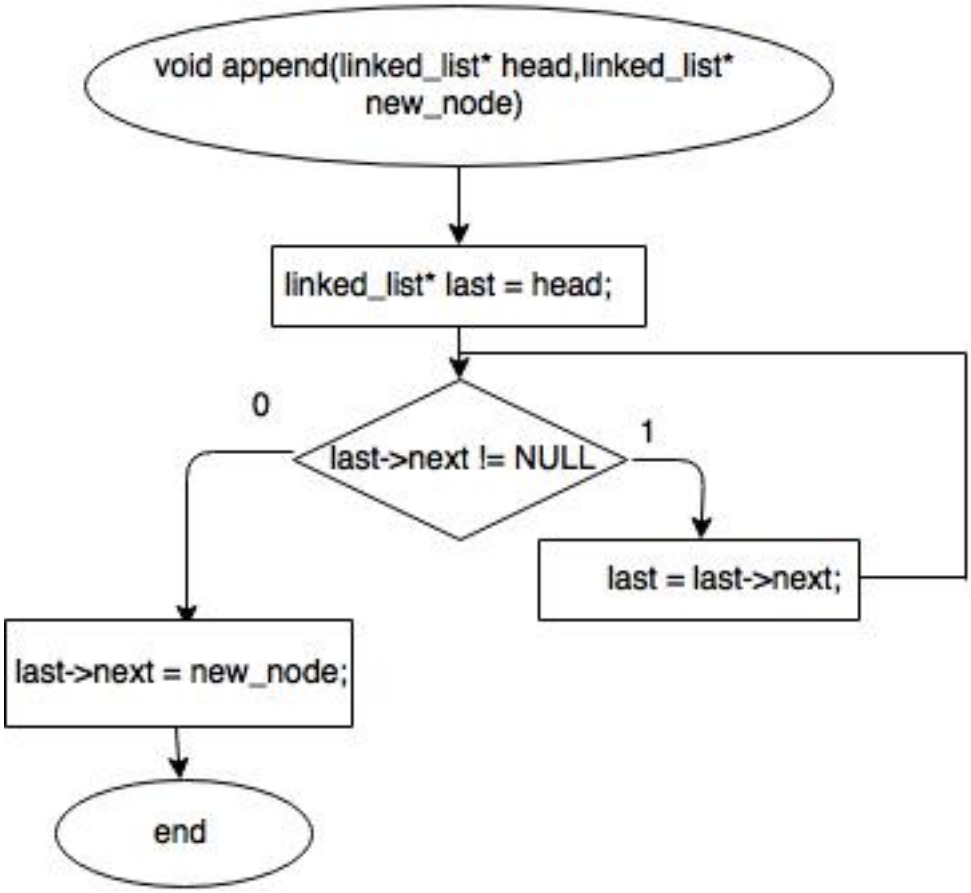
Перевірив:
Караюз І.В

Київ – 2017

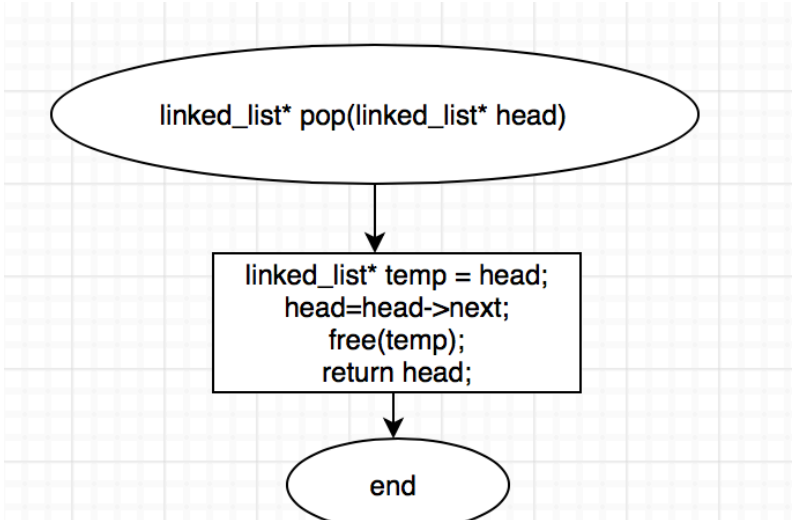
Завдання: для однозв’язного списку груп реалізувати :
А) Додавання елемента у хвіст списку.
Б)Видалення елемента з голови списку.
В)Поміняти місцями елемент з найменшою кількістю студентів с елементом у хвості.
Г)Надрукувати весь список.
Д)Видалити весь список.

Блок – схеми реалізованих функцій:

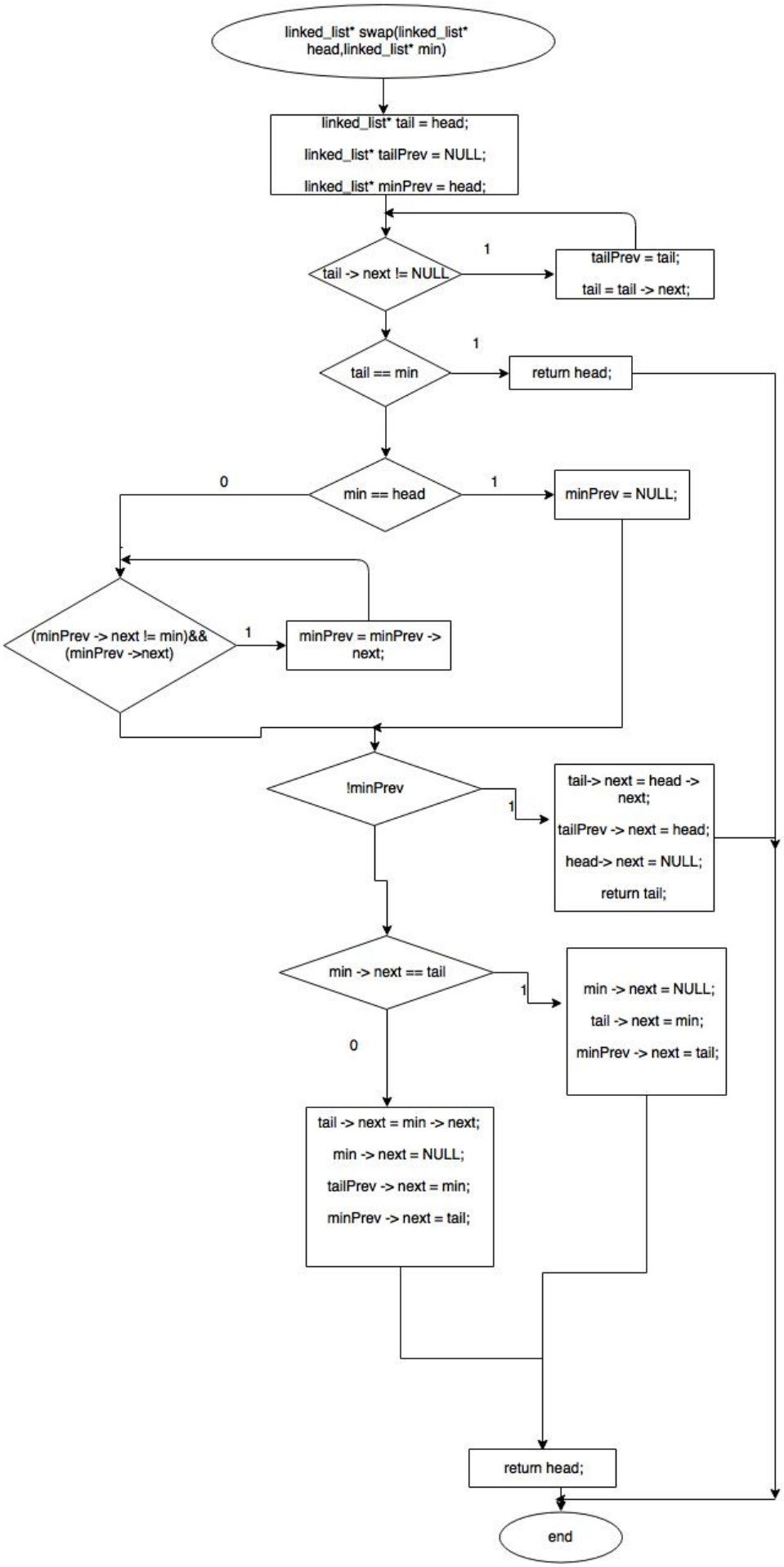
А) Додавання елемента у хвіст списку

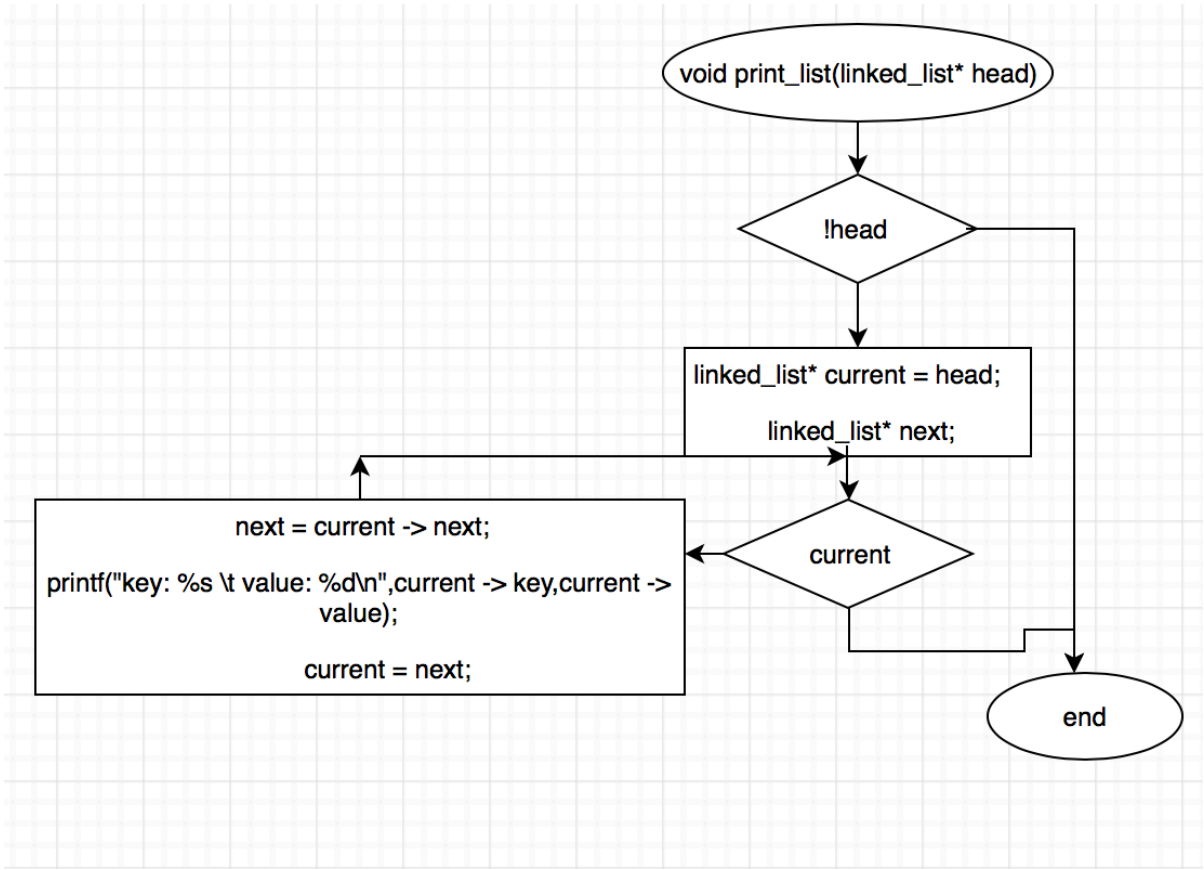


Б) Видалення елемента з голови списку

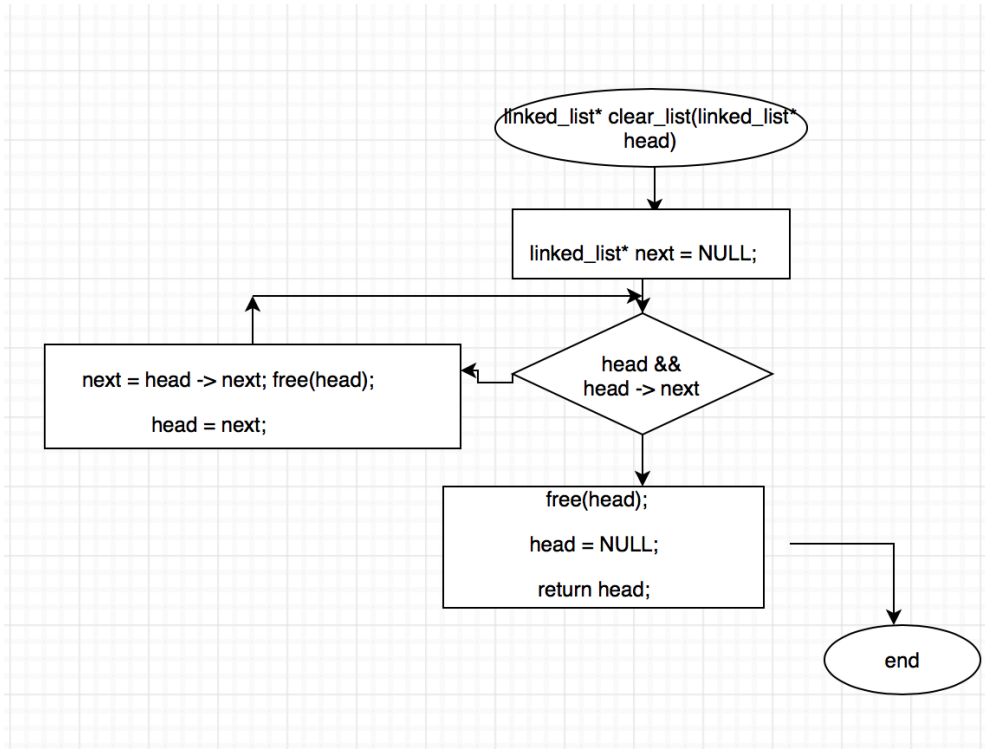


хвості.


linked_list* swap(linked_list* head,linked_list* min)
linked_list* tail = head;
linked_list* tailPrev = NULL;
linked_list* minPrev = head;
tail -> next != NULL
1
tailPrev = tail;
tail = tail -> next;
tail == min
1
return head;
min == head
0
1
minPrev = NULL;
(minPrev -> next != min)&& (minPrev ->next)
1
minPrev = minPrev -> next;
!minPrev
1
tail-> next = head -> next;
tailPrev -> next = head;
head-> next = NULL;
return tail;
min -> next == tail
1
min -> next = NULL;
tail -> next = min;
minPrev -> next = tail;
0
tail -> next = min -> next;
min -> next = NULL;
tailPrev -> next = min;
minPrev -> next = tail;
return head;
end



Д)Видалити весь список.



Лістинг програми:

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

typedef struct list{
    int value;
    char* key;
    struct list* next;

}linked_list;

void add(linked_list* current_node,linked_list* new_node);
linked_list* create_node(int value,char* key);
linked_list* pop(linked_list* head);
void print_list(linked_list* head);
void append(linked_list* head, linked_list* new_node);
linked_list* clear_list(linked_list* head);
linked_list* min(linked_list* head);
linked_list* swap(linked_list* head,linked_list* min);
int node_cmp(linked_list* a, linked_list* b);

int main(void) {

    linked_list* head = (linked_list*)malloc(sizeof(linked_list));
    head -> value = 1;
    head -> key = "DA72";
    head -> next = NULL;

    linked_list* new_node1 = create_node(20,"DA71");
    append(head, new_node1);
    linked_list* new_node2 = create_node(10,"DA70");
    add(head, new_node2);
    linked_list* new_node3 = create_node(5,"DA64");
    add(head, new_node3);
    print_list(head);
    head = pop(head);
    printf("\n after deleting the head node:\n");
    print_list(head);
    printf("\n after swapping min and tail:\n");
    head = swap(head, min(head));
    print_list(head);

    head = clear_list(head);
    print_list(head);


}

linked_list* create_node(int value,char* key){
    linked_list* new_node = (linked_list*)malloc(sizeof(linked_list));
    if(!new_node){
        printf("ERROR.CREATE_NODE : BAD ALLOCATION");
        exit(1);
    };
    new_node -> value = value;
    new_node -> key = key;
    new_node -> next = NULL;
    return new_node;
```

```
void add(linked_list* current_node,linked_list* new_node){
    new_node -> next = current_node -> next;
    current_node -> next = new_node;
    return;
}

linked_list* clear_list(linked_list* head){
       int count = 0;
    if (!head) {
        printf("list is empty \n\n");
        return 0;
    }
    linked_list* next = NULL;
        while(head && head -> next){
            next = head -> next;
            free(head);
            head = next;

            count++;
        }
    free(head);
    head = NULL;

     //  head = NULL;
    printf("list is clear \n\n");

    return head;
}
linked_list* pop(linked_list* head){
    linked_list* temp = head ;
    head = head -> next;
    free(temp);
    return head;
}

void print_list(linked_list* head){
    if(!head){
     printf("list is empty\n");
        return;
    }
    linked_list* current = head;
    linked_list* next;
    while(current != NULL){
        next = current -> next;
        printf("key: %s \t value: %d\n",current -> key,current -> value);
        current = next;
    }
    printf("\n\n");
}

void append(linked_list* head,linked_list* new_node){

    linked_list* last = head;

    while(last -> next != NULL){
        last = last -> next;
    }
    last -> next = new_node;

}
```

```
    linked_list* min = head;
    while(current != NULL){
        if(node_cmp(current, min)){
            min = current;
        }
        current = current -> next;
    }
    printf("minKey: %s \t minVal: %d \n\n", min -> key, min -> value);
    return min;
}

linked_list* swap(linked_list* head,linked_list* min){
    linked_list* tail = head;
    linked_list* tailPrev = NULL;
    linked_list* minPrev = head;

  while (tail -> next != NULL){
        tailPrev = tail;
        tail = tail -> next;
  }
    if(tail == min ) return head ;
    if(min == head){
        minPrev = NULL;
    }else{

   while((minPrev -> next != min)&&(minPrev ->next)){
        minPrev = minPrev -> next;
   }
    }
    if(!minPrev){
        tail-> next = head -> next;
        tailPrev -> next = head;
        head-> next = NULL;
        return tail;
    }
    if(min -> next == tail){
        min -> next = NULL;
        tail -> next = min;
        minPrev -> next = tail;
    }else{
        tail -> next = min -> next;
        min -> next = NULL;
        tailPrev -> next = min;
        minPrev -> next = tail;
        }
    return head;
}

int node_cmp(linked_list* a, linked_list* b){ //a - current value ; b -
current min value

    if (strcmp(a -> key, b -> key)==0) {
        if (a ->value < b -> value) {
            return 1;
        }else return 0;
    }
    if (strcmp(a -> key, b -> key) < 0){
        return 1;
    }
    if (strcmp(a -> key, b -> key) > 0){
        return 0;
```

Висновки: У ході виконання комп’ютерного практикуму мовою С був написаний код, що ініціалізує однозв’язний список, реалізує п’ять функцій: додавання елемента у хвіст списку, видалення елемента з голови списку, зміну місць елемента у кінці списку з елементом з найменшою кількістю студентів, вивід та видалення всього списку.