מקרא נקודות - מבחן מועד א' תשפ"ב
תכנות מערכות א'

להלן הורדת הנקודות על טעויות נפוצות.
אם הקוד לא מתקמפל לפחות חצי מהנק' של השאלה יורדות

שאלה 1 (33 נק')

א. קליטה של מחרוזת

a. **מינוס 8:** קליטת אורך המחרוזת\הגבלת גודל מחרוזת בקבוע

b. **מינוס 5:** אי עצירה נכונה של הקלט, הקצאה לא נכונה, שחרור לא נכון, כל טעות אחרת

מימוש:

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

char* input() {
   int count = 0;
   char c;
   char* str = (char*)malloc((count+1)*sizeof(char));
   char* cpy = str;
   while(scanf("%c",&c) && c != '\n'){
      count++;
      str = realloc(str, (count+1)*sizeof(char));
      if(!str){
         printf("Too long input");
         free(cpy);
         return NULL;
      }
      cpy = str;
      str[count-1] = c;
   }
   str[count] = '\0';
   return str;
}

int main(){
   char* str = input();
   printf("%s\n", str);
   free(str);
   return 0;
}
```

ב.

מימוש שגוי של אחד הפונקציות (לכל פונק')– מינוס 4
אין בדיקת הצלחה של הקצאה – מינוס 5
תנאי לא נכון לבדיקה אם התור מלא \ריק – מינוס 3
מיין לא נכון או לא תואם מימוש – מינוס 6
מימוש מחסנית – מינוס 8
פתרון לא יעיל בעליל – מינוס 3
תור חד פעמי. אי אפשר למלא ולרוקן ושוב למלא - מינוס 5

ג. <u>התשובה:</u>

i = 5, j = 10
i = 31, j = 5

מינוס 5: כל שגיאה בהדפסה

<u>שאלה 2 (33 נק')</u>

א. (1) sizeof(m)= 7*sizeof(char*) sizeof(*m)= sizeof(char *)
(2) sizeof(void*) 7*sizeof(char)

כל טעות – מינוס 5, מקסימום הורדה 11 נקודות

ב. **מינוס 4**:השוואת תווים ולא מילים
מינוס 6:עבודה עם מחרוזות ולא קבצים
מינוס 8: עבודה לא נכונה עם קבצים, לוגיקה לא נכונה, מימוש שלא עובד
מינוס 5: כל טעות אחרת

ג. **מינוס 4:** הגבלת גודל מחרוזת, טעות קטנה שאינה מונעת מקוד לעבוד
מינוס 6 : טעות לוגית, מימוש לא מדוייק אבל עובד

שאלה 3 (34 נק')

מינוס 8 על מימוש חסר או לא עובד, מינוס 5 על כל טעות אחרת כולל מימוש חלקי שעובד:

יצירת קודקוד , אין בדיקת הצלחה של הקצאה, הכנסה לרשימה , מיון, הדפסת לרשימה חסר\שגוי, היפוך חסר\שגוי, היפוך לא מלא, שחרור שגוי

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

typedef struct n{
   int val;
   struct n * next;
} node;

node * create(int v){
   node * new = (node*)malloc(sizeof(node));
   if(!new)
      return NULL;
   new->val = v;
   new->next = NULL;
   return new;
}

int isDesc(node* h){
   int flag = 0;
   while(h->next){
      if(h->val>h->next->val)
         if(flag==1)
            return 0;
         else flag = -1;
      if(h->val<h->next->val)
         if(flag==-1)
            return 0;
         else flag = 1;
      h = h->next;
   }
   return flag;
}
```

```
void add(node** h, int v){
   node* new = create(v);
   if(!h||!new)
      return;
   if(!(*h)){
      *h = new;
      return;
   }
   if(!(*h)->next){
      (*h)->next=new;
      return;
   }
   int flag = isDesc(*h);
   if(flag*(*h)->val>flag*v){
      new->next = *h;
      *h = new;
      return;
   }
   node* tmp = *h;
   while(tmp->next && flag*tmp->next->val<flag*v)
      tmp = tmp->next;
   new->next = tmp->next;
   tmp->next=new;
}

void print(node* h, int v){
   while(h){
      if(h->val >= v)
         printf("%d ->", h->val);
      h=h->next;
   }
   printf("\n");
}

void reverse(node** h){
   if(!h || !(*h)) return;
```

```
    node* curr = *h;
    node* next = curr->next;
    curr->next = NULL;
    while(next){
        node* tmp = next;
        next = next->next;
        tmp->next = curr;
        curr = tmp;
    }
    *h = curr;
}

void freeList(node** h){
    while(h && *h){
        node *tmp = *h;
        *h = (*h)->next;
        free(tmp);
    }
}

int main(){
    node* head = NULL;
    add(&head, 3);
    add(&head, 2);
    add(&head, 5);
    add(&head, 1);
    add(&head, 7);
    add(&head, 0);
    add(&head, 4);
    add(&head, 6);
    print(head, 0);
    reverse(&head);
    print(head,0);
    freeList(&head);
    return 0;
}
```