The next ujbnu will be directly imported from a file

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

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

void swap(int *a, int *b) {
  int temp=*a;
  *a=*b;
  *b=temp;
}

/* Genera in memoria dinamica un array ordinato dei primi n numeri
    naturali.
 * Ritorna il puntatore al primo elemento */
int* seqgen(int n){
  int* v = (int*)malloc(n*sizeof(*v));
  for (int i = 0; i < n; i++) {
    v[i] = i;
  }
  return v;
}

void randomize(int* r, int n){
  srand(time(NULL));
  for (int i = n-1; i >= 0; --i){
      int j = rand() % (i+1);
      int temp = r[i];
      r[i] = r[j];
      r[j] = temp;
  }
}

int partition(int a[], int n) {
  int k = 1;
  for(int i = 1; i<n; i++)
    if (a[i] < a[0])
      swap(&a[i], &a[k++]);
  swap(&a[0], &a[k-1]);
  return k-1;
}

void quicksort(int a[], int n) {
  int k;
  if (n<2)
    return;
  k = partition(a, n);
  quicksort(a, k);
  quicksort(a+k+1, n-k-1);
}
```

```c
int linsearch(int a[], int n, int el){
  for (int i = 0; i < n; i++) {
    if(a[i] == el){
      return i;
    }
  }
  return -1;
}

int binsearch (int a[], int n, int x) {
  int first = 0;
  int last = n - 1;
  int chosen = (first + last) / 2;
  while (first <= last) {
    if (a[chosen] == x)
      return chosen;
    else if (a[chosen] < x)
      first = chosen + 1;
    else
      last = chosen - 1;
    chosen = (first + last) / 2;
  }
  return -1;
}

typedef struct node node;
typedef node* tree;

struct node{
  int* pivot;
  int* data;
  int size;
  tree left;
  tree right;
};

tree newtree(int a[], int n){
  node* new;
  new = (node *) malloc(sizeof(node));
  new -> pivot = NULL;
  new -> data = a;
  new -> size = n;
  new -> left = NULL;
  new -> right = NULL;

  //printf("newtree: ");
  //print(a, n);
  return new;
}

int* sorch (tree t, int x){
  if (t == NULL) return NULL;
  if (t->pivot == NULL){
    int pivot = t->data[0];
    int smaller = x < pivot;
    int* found = NULL;
    int k = 1;
```

```
    for(int i = 1; i < t->size; i++){
      if (t->data[i] < pivot)
        swap(&t->data[i], &t->data[k++]);
        //printf("swapped\n");
      if(smaller){
        if(t->data[k-1] == x)
          found = &t->data[k-1];
      } else {
        if(t->data[i] == x)
          found = &t->data[i];
```