הטכניון – הפקולטה למדעי המחשב
מבוא למדמ"ח מ'/ח' 234114/117
פתרון מועד א' סמסטר אביב תשע"א

שאלה 1

סעיף א
סיבוכיות זמן: $\Theta(n^2)$
סיבוכיות מקום: $\Theta(n)$
בלוק הקוד ידפיס את התו '2'

סעיף ב
סיבוכיות זמן: $\Theta(n^2 \log n)$
סיבוכיות מקום: $\Theta(1)$
שינוי תנאי בדיקה: **i * i < n**

שאלה 2

```
int handle_text(char* st)
{
  char* out = st;
  int count = 0;
  while (*st) {
    *out = *st;
    out++;
    st++;
    if (*(st-1) == ' ') {
      while (*st == ' ') {
        st++;
        count++;
      }
    }
  }
  *out = '\0';
  return count;
}
```

שאלה 3

להלן פיתרון בסיבוכיות זמן $\Theta(n)$ ומקום $\Theta(\log n)$

```
/* The recursive function returns both
   the required function and the average
   of the array, which is recursively
   computed as the average of the left
   and right averages.
*/
int check_sorted_in_averages_helper(
  int a[], int n, double* average);

int check_sorted_in_averages(int a[], int n)
{
  double average;
  return check_sorted_in_averages_helper(a, n, &average);
}

int check_sorted_in_averages_helper(
  int a[], int n, double* average)
{
  int result_left, result_right;
  double av_left, av_right;

  if (n == 2) {
    *average = (a[0] + a[1]) / 2.0;
    return a[0] <= a[1];
  }

  result_left =
    check_sorted_in_averages_helper(a, n/2, &av_left);
  result_right =
    check_sorted_in_averages_helper(a + n/2, n/2, &av_right);
  *average = (av_left + av_right) / 2.0;
  return result_left && result_right && (av_left <= av_right);
}
```

שאלה 4

```
void find_endwords_helper(
  Word dictionary[], int dictionary_size, int length,
  int i, Word words[], int count_love);

void find_endwords(
  Word dictionary[], int dictionary_size, int length)
{
  Word* words = malloc(length * sizeof(Word));
  find_endwords_helper(
    dictionary, dictionary_size, length, 0, words, 0);
  free(words);
}

void find_endwords_helper(
  Word dictionary[], int dictionary_size, int length,
  int i, Word words[], int count_love)
{
  int j;

  if (i == length) {
    if (count_love >= 2 && count_love <= 4)
      print_words(words, length);
    return;
  }

  for (j = 0; j < dictionary_size; j++) {

    if (count_love == 4 && is_love(dictionary[j]))
      continue;

    if ((i==0) ||
        (i % 2) && is_rhyme(dictionary[j], words[i - 1]) ||
        !(i % 2) && (i > 1) &&
          !is_rhyme(dictionary[j], words[i - 1])) {
      words[i] = dictionary[j];
      find_endwords_helper(
        dictionary, dictionary_size,
        length, i + 1,  words,
        count_love + is_love(dictionary[j]));
    }
  }
}
```