

Міністерство освіти і науки України
Національний технічний університет України «КПІ ім. Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Мультипарадигменне програмування

ЗВІТ

до лабораторної роботи №1

Виконав студент ІІ-01 Коваленко Микита

Прийняв ас. Очеретяний О. К.

Київ 2021

1. Завдання лабораторної роботи

Завдання 1:

Обчислювальна задача тут тривіальна: для текстового файлу ми хочемо відобразити N (наприклад, 25) найчастіших слів і відповідну частоту їх повторення, упорядковано за зменшенням. Слід обов'язково нормалізувати використання великих літер і ігнорувати стоп-слова, як «the», «for» тощо. Щоб все було просто, ми не піклуємося про порядок слів з однаковою частотою повторень. Ця обчислювальна задача відома як term frequency.

Ось такий вигляд матимуть ввід і відповідно вивід результату програми:

Input:

White tigers live mostly in India

Wild lions live mostly in Africa

Output:

live - 2

mostly - 2

africa - 1

india - 1

lions - 1

tigers - 1

white - 1

wild - 1

Завдання 2:

Тепер, нам потрібно виконати задачу, що називається словниковим індексуванням. Для текстового файлу виведіть усі слова в алфавітному порядку разом із номерами сторінок, на яких Ці слова знаходяться. Ігноруйте всі слова, які зустрічаються більше 100 разів. Припустимо, що сторінка являє собою послідовність із 45 рядків.

Наприклад, якщо взяти книгу Pride and Prejudice, перші кілька записів індексу будуть:

abatement - 89

abhorrence - 101, 145, 152, 241, 274, 281

abhorrent - 253

abide - 158, 292

2. Опис алгоритму

Завдання 1

1. Зчитати всі рядки з текстового файлу
2. Зчитати наступне слово, привівши всі літери до нижнього регістру. Якщо слів не залишилося - перейти до кроку 6

3. Якщо слово є стоп-словом - перейти до кроку 2
4. Якщо масив повторень містить зчитане слово - збільшити кількість повторень на 1, інакше - додати слово до масиву
5. Перейти до кроку 2
6. Відсортувати масив алгоритмом бульбашки
7. Записати перші 25 слів відсортованого масиву до вихідного файлу

Завдання 2

1. Зчитати всі рядки з текстового файлу
2. Зчитати наступне слово, привівши всі літери до нижнього регістру. Якщо слів не залишилося - перейти до кроку 6
3. Знайти сторінку, на якій зустрілося слово
4. Якщо слово вже є в масиві слів, а сторінки нема - додати сторінку до масиву сторінок
5. Якщо слова нема в масиві слів - додати слово до масиву
6. Відсортувати масив слів в алфавітному порядку
7. Обрати наступне слово з масиву відсортованих слів. Якщо слів не залишилося - перейти до кроку 9
8. Якщо слово зустрілося менше 100 разів - записати слово й сторінки з повтореннями у файл. Перейти до кроку 7
9. Кінець алгоритму

3. Вихідний код

Завдання 1

```
using System;
using System.IO;

namespace task1
{
    public class Program
    {
        public static void Main(string[] args)
        {
            string[] lines = File.ReadAllLines(args[0]);

            string[] stopWords =
            {
                "i", "me", "my", "myself", "we", "our", "ours",
                "ourselves", "you", "your", "yours", "yourself", "yourselves", "he",
                "him", "his", "himself", "she", "her", "hers", "herself", "it", "its",
                "itself", "they", "them", "their", "theirs", "themselves", "what",
                "which", "who", "whom", "this", "that", "these", "those", "am", "is",
                "are", "was", "were", "be", "been", "being", "have", "has", "had",
                "having", "do", "does", "did", "doing", "a", "an", "the", "and", "but",
                "if", "or", "because", "as", "until", "while", "of", "at", "by", "for",
                "with", "about", "against", "between", "into", "through", "during",
                "before", "after", "above", "below", "to", "from", "up", "down", "in",
                "out", "on", "off", "over", "under", "again", "further", "then",
                "once", "here", "there", "when", "where", "why", "how", "all", "any",
```

```

"both", "each", "few", "more", "most", "other", "some", "such", "no",
"nor", "not", "only", "own", "same", "so", "than", "too", "very", "s",
"t", "can", "will", "just", "don", "should", "now"
};

```

```

(string value, int count)[] wordOccurrences =
Array.Empty<(string, int)>();

```

```

int lineIndex = 0;

```

```

lines_loop:
if (lines.Length == 0)
{
    goto bubble_sort;
}

```

```

string currentLine = lines[lineIndex];
currentLine += ' ';

```

```

int charIndex = 0;

```

```

int wordStartIndex = 0;

```

```

line_loop:
if (currentLine.Length <= 1)
{
    goto line_loop_end;
}

```

```

char currentChar = currentLine[charIndex];

```

```

if (currentChar is ' ')
{

```

```

    string word = "";

```

```

    int wordCharIndex = wordStartIndex;

```

```

word_copy:

```

```

    char? currentWordChar = currentLine[wordCharIndex];

```

```

    if (currentWordChar is >= 'A' and <= 'Z')
    {

```

```

        currentWordChar = (char)(currentWordChar + 32);
    }

```

```

    if (currentWordChar is < 'a' or > 'z')
    {

```

```

        currentWordChar = null;
    }

```

```

    word += currentWordChar;

```

```

    wordCharIndex++;

```

```

    if (wordCharIndex < charIndex)
    {

```

```

        goto word_copy;
    }

```

```

        if (word == string.Empty)
        {
            goto line_loop_end;
        }

        int stopWordIndex = 0;

stopword_check:

        string stopWord = stopWords[stopWordIndex];

        if (stopWord == word)
        {
            wordStartIndex = charIndex + 1;
            goto line_loop_end;
        }

        stopWordIndex++;

        if (stopWordIndex < stopWords.Length)
        {
            goto stopword_check;
        }

        int occurrenceIndex = 0;

occurrence_check:

        if (occurrenceIndex < wordOccurrences.Length &&
wordOccurrences[occurrenceIndex].value != word)
        {
            occurrenceIndex++;
            goto occurrence_check;
        }

        if (occurrenceIndex < wordOccurrences.Length)
        {
            wordOccurrences[occurrenceIndex].count++;
        }
        else
        {
            (string, int)[] newOccurrences = new (string,
int)[wordOccurrences.Length + 1];

            int newOccurrencesIndex = 0;

newOccurrences_loop:
            if (newOccurrencesIndex >= wordOccurrences.Length)
            {
                goto newOccurrenceAdd;
            }

            newOccurrences[newOccurrencesIndex] =
wordOccurrences[newOccurrencesIndex];
            newOccurrencesIndex++;

            if (newOccurrencesIndex < wordOccurrences.Length)
            {
                goto newOccurrences_loop;
            }

```

```

        newOccurrenceAdd:

            newOccurrences[newOccurrencesIndex] = (word, 1);

            wordOccurrences = newOccurrences;
        }

        wordStartIndex = charIndex + 1;
    }

line_loop_end:

    charIndex++;

    if (charIndex < currentLine.Length)
    {
        goto line_loop;
    }

    lineIndex++;

    if (lineIndex < lines.Length)
    {
        goto lines_loop;
    }

bubble_sort:

    int i = 0;
    int j = 0;

outer_sort:

inner_sort:

    if (j >= wordOccurrences.Length - 1)
    {
        goto inner_loop_end;
    }

    var curr = wordOccurrences[j];
    var next = wordOccurrences[j + 1];

    if (curr.count < next.count)
    {
        wordOccurrences[j + 1] = curr;
        wordOccurrences[j] = next;
    }

inner_loop_end:

    j++;

    if (j < wordOccurrences.Length - 1)
    {
        goto inner_sort;
    }

    j = 0;

```

```

        i++;

        if (i < wordOccurrences.Length - 1)
        {
            goto outer_sort;
        }

        using StreamWriter writer = new("occurrences.txt");
        int occurrence_index = 0;

    writer_loop:
        if (occurrence_index >= wordOccurrences.Length)
        {
            goto writer_loop_end;
        }

        var occurrence = wordOccurrences[occurrence_index];

        writer.WriteLine($"{occurrence.value} {occurrence.count}");

    writer_loop_end:

        occurrence_index++;

        if (occurrence_index < wordOccurrences.Length &&
occurrence_index < 25)
        {
            goto writer_loop;
        }
    }
}

```

Завдання 2

```

using System;
using System.IO;

namespace task2
{
    public class Program
    {
        public static void Main(string[] args)
        {
            string[] lines = File.ReadAllLines(args[0]);

            (string value, int count, int[] pages)[] wordOccurrences =
Array.Empty<(string, int, int[])>();

            int lineIndex = 0;

        lines_loop:
            if (lines.Length == 0)
            {
                goto bubble_sort;
            }

            string currentLine = lines[lineIndex];
            currentLine += ' ';

            int charIndex = 0;

```

```

        int wordStartIndex = 0;

line_loop:
    if (currentLine.Length <= 1)
    {
        goto line_loop_end;
    }

    char currentChar = currentLine[charIndex];

    if (currentChar is ' ')
    {
        string word = "";

        int wordCharIndex = wordStartIndex;

word_copy:

        char? currentWordChar = currentLine[wordCharIndex];

        if (currentWordChar is >= 'A' and <= 'Z')
        {
            currentWordChar = (char)(currentWordChar + 32);
        }

        if (currentWordChar is < 'a' or > 'z')
        {
            currentWordChar = null;
        }

        word += currentWordChar;

        wordCharIndex++;

        if (wordCharIndex < charIndex)
        {
            goto word_copy;
        }

        int occurrenceIndex = 0;

occurrence_check:

        if (occurrenceIndex < wordOccurrences.Length &&
wordOccurrences[occurrenceIndex].value != word)
        {
            occurrenceIndex++;
            goto occurrence_check;
        }

        if (occurrenceIndex < wordOccurrences.Length)
        {
            ref var wordOccurrence = ref
wordOccurrences[occurrenceIndex];
            wordOccurrence.count++;
            int currentPageNumber = lineIndex / 45 + 1;

            bool pageExists = false;
            int pageCheckIndex = 0;

```



```

        page_check:
            if (pageCheckIndex >= wordOccurrence.pages.Length)
            {
                goto page_check_next;
            }

            if (currentPageNumber ==
wordOccurrence.pages[pageCheckIndex])
            {
                pageExists = true;
                goto page_check_next;
            }

            pageCheckIndex++;

            if (pageCheckIndex < wordOccurrence.pages.Length)
            {
                goto page_check;
            }

        page_check_next:
            if (!pageExists)
            {
                int[] newPages = new int[wordOccurrence.pages.Length
+ 1];

                int newPagesIndex = 0;

                add_page:

                    if (newPagesIndex >= wordOccurrence.pages.Length)
                    {
                        goto add_new_page;
                    }

                    newPages[newPagesIndex] =
wordOccurrence.pages[newPagesIndex];

                    newPagesIndex++;

                    if (newPagesIndex < wordOccurrence.pages.Length)
                    {
                        goto add_page;
                    }

                add_new_page:
                    newPages[^1] = currentPageNumber;
                    wordOccurrence.pages = newPages;
            }
        else
        {
            (string, int, int[])[] newOccurrences = new (string, int,
int[])[wordOccurrences.Length + 1];

            int newOccurrencesIndex = 0;

            newOccurrences_loop:
                if (newOccurrencesIndex >= wordOccurrences.Length)
                {

```

```

        goto newOccurrenceAdd;
    }

    newOccurrences[newOccurrencesIndex] =
wordOccurrences[newOccurrencesIndex];
    newOccurrencesIndex++;

    if (newOccurrencesIndex < wordOccurrences.Length)
    {
        goto newOccurrences_loop;
    }

    newOccurrenceAdd:

        newOccurrences[newOccurrencesIndex] = (word, 1, new[] {
lineIndex / 45 + 1 });

        wordOccurrences = newOccurrences;
    }

    wordStartIndex = charIndex + 1;
}

line_loop_end:

    charIndex++;

    if (charIndex < currentLine.Length)
    {
        goto line_loop;
    }

    lineIndex++;

    if (lineIndex < lines.Length)
    {
        goto lines_loop;
    }

bubble_sort:

    int i = 0;
    int j = 0;

outer_sort:

inner_sort:
    if (j >= wordOccurrences.Length - 1)
    {
        goto inner_loop_end;
    }

    var curr = wordOccurrences[j];
    var next = wordOccurrences[j + 1];

    int wordCompareIndex = 0;
    bool shouldSwap = false;
    int minLength = curr.value.Length > next.value.Length ?
next.value.Length : curr.value.Length;

```

```

word_compare:

if (wordCompareIndex >= minLength)
{
    goto word_compare_end;
}

char currChar = curr.value[wordCompareIndex];
char nextChar = next.value[wordCompareIndex];

if (currChar > nextChar)
{
    shouldSwap = true;
    goto word_compare_end;
}

if (currChar < nextChar)
{
    goto word_compare_end;
}

wordCompareIndex++;

if (wordCompareIndex < curr.value.Length || wordCompareIndex <
next.value.Length)
{
    goto word_compare;
}

word_compare_end:

if (!shouldSwap && wordCompareIndex == minLength &&
curr.value.Length > next.value.Length)
{
    shouldSwap = true;
}

if (shouldSwap)
{
    wordOccurrences[j + 1] = curr;
    wordOccurrences[j] = next;
}

inner_loop_end:

j++;

if (j < wordOccurrences.Length - 1)
{
    goto inner_sort;
}

j = 0;
i++;

if (i < wordOccurrences.Length - 1)
{
    goto outer_sort;
}

```

```

        using StreamWriter writer = new("occurrences.txt");
        int occurrence_index = 0;

writer_loop:
        if (occurrence_index >= wordOccurrences.Length)
        {
            goto writer_loop_end;
        }

        var occurrence = wordOccurrences[occurrence_index];

        if (occurrence.count >= 100)
        {
            goto writer_loop_end;
        }

        string pagesString = string.Empty;

        int currentPageIndex = 0;

        page_string_loop:

        if (currentPageIndex >= occurrence.pages.Length)
        {
            goto write_pages;
        }

        pagesString += $"{occurrence.pages[currentPageIndex]}";

        if (currentPageIndex < occurrence.pages.Length - 1)
        {
            pagesString += ", ";
        }

        currentPageIndex++;

        if (currentPageIndex < occurrence.pages.Length)
        {
            goto page_string_loop;
        }

        write_pages:

        writer.WriteLine($"{occurrence.value} - {pagesString}");

writer_loop_end:

        occurrence_index++;

        if (occurrence_index < wordOccurrences.Length)
        {
            goto writer_loop;
        }
    }
}

```

4. Приклади роботи

Mr 783

Elizabeth 593

could 525

would 468

said 402

Darcy 371

Mrs 344

much 328

must 318

Bennet 294

Miss 283

one 269

Jane 262

Bingley 257

know 239

though 221

never 220

soon 215

well 213

think 211

may 207

might 201

time 200

every 197

little 189

abatement - 99

abhorrence - 111, 160, 167, 263, 299, 306

abhorrent - 276

abide - 174, 318

abiding - 177

abilities - 72, 74, 107, 155, 171, 194

able - 19, 37, 58, 78, 84, 86, 88, 91, 98, 101, 107, 109, 110, 120, 126, 130, 131, 144, 145, 152, 156, 172, 174, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 250, 251, 252, 253, 254, 255, 256, 257, 258, 259, 260, 261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 271, 272, 273, 274, 275, 276, 277, 278, 279, 280, 281, 282, 283, 284, 285, 286, 287, 288, 289, 290, 291, 292, 293, 294, 295, 296, 297, 298, 299, 300, 301, 302, 303, 304, 305, 306, 307, 308, 309, 310, 311, 312, 313, 314, 315, 316, 317, 318, 319, 320, 321, 322, 323, 324, 325, 326, 327, 328, 329, 330, 331, 332, 333, 334, 335, 336, 337, 338, 339, 340, 341, 342, 343, 344, 345, 346, 347, 348, 349, 350, 351, 352, 353, 354, 355, 356, 357, 358, 359, 360, 361, 362, 363, 364, 365, 366, 367, 368, 369, 370, 371, 372, 373, 374, 375, 376, 377, 378, 379, 380, 381, 382, 383, 384, 385, 386, 387, 388, 389, 390, 391, 392, 393, 394, 395, 396, 397, 398, 399, 400, 401, 402, 403, 404, 405, 406, 407, 408, 409, 410, 411, 412, 413, 414, 415, 416, 417, 418, 419, 420, 421, 422, 423, 424, 425, 426, 427, 428, 429, 430, 431, 432, 433, 434, 435, 436, 437, 438, 439, 440, 441, 442, 443, 444, 445, 446, 447, 448, 449, 450, 451, 452, 453, 454, 455, 456, 457, 458, 459, 460, 461, 462, 463, 464, 465, 466, 467, 468, 469, 470, 471, 472, 473, 474, 475, 476, 477, 478, 479, 480, 481, 482, 483, 484, 485, 486, 487, 488, 489, 490, 491, 492, 493, 494, 495, 496, 497, 498, 499, 500, 501, 502, 503, 504, 505, 506, 507, 508, 509, 510, 511, 512, 513, 514, 515, 516, 517, 518, 519, 520, 521, 522, 523, 524, 525, 526, 527, 528, 529, 530, 531, 532, 533, 534, 535, 536, 537, 538, 539, 540, 541, 542, 543, 544, 545, 546, 547, 548, 549, 550, 551, 552, 553, 554, 555, 556, 557, 558, 559, 560, 561, 562, 563, 564, 565, 566, 567, 568, 569, 570, 571, 572, 573, 574, 575, 576, 577, 578, 579, 580, 581, 582, 583, 584, 585, 586, 587, 588, 589, 590, 591, 592, 593, 594, 595, 596, 597, 598, 599, 600, 601, 602, 603, 604, 605, 606, 607, 608, 609, 610, 611, 612, 613, 614, 615, 616, 617, 618, 619, 620, 621, 622, 623, 624, 625, 626, 627, 628, 629, 630, 631, 632, 633, 634, 635, 636, 637, 638, 639, 640, 641, 642, 643, 644, 645, 646, 647, 648, 649, 650, 651, 652, 653, 654, 655, 656, 657, 658, 659, 660, 661, 662, 663, 664, 665, 666, 667, 668, 669, 670, 671, 672, 673, 674, 675, 676, 677, 678, 679, 680, 681, 682, 683, 684, 685, 686, 687, 688, 689, 690, 691, 692, 693, 694, 695, 696, 697, 698, 699, 700, 701, 702, 703, 704, 705, 706, 707, 708, 709, 710, 711, 712, 713, 714, 715, 716, 717, 718, 719, 720, 721, 722, 723, 724, 725, 726, 727, 728, 729, 730, 731, 732, 733, 734, 735, 736, 737, 738, 739, 740, 741, 742, 743, 744, 745, 746, 747, 748, 749, 750, 751, 752, 753, 754, 755, 756, 757, 758, 759, 760, 761, 762, 763, 764, 765, 766, 767, 768, 769, 770, 771, 772, 773, 774, 775, 776, 777, 778, 779, 780, 781, 782, 783, 784, 785, 786, 787, 788, 789, 790, 791, 792, 793, 794, 795, 796, 797, 798, 799, 800, 801, 802, 803, 804, 805, 806, 807, 808, 809, 810, 811, 812, 813, 814, 815, 816, 817, 818, 819, 820, 821, 822, 823, 824, 825, 826, 827, 828, 829, 830, 831, 832, 833, 834, 835, 836, 837, 838, 839, 840, 841, 842, 843, 844, 845, 846, 847, 848, 849, 850, 851, 852, 853, 854, 855, 856, 857, 858, 859, 860, 861, 862, 863, 864, 865, 866, 867, 868, 869, 870, 871, 872, 873, 874, 875, 876, 877, 878, 879, 880, 881, 882, 883, 884, 885, 886, 887, 888, 889, 890, 891, 892, 893, 894, 895, 896, 897, 898, 899, 900, 901, 902, 903, 904, 905, 906, 907, 908, 909, 910, 911, 912, 913, 914, 915, 916, 917, 918, 919, 920, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000

ablution - 119

abode - 59, 60, 66, 110, 122, 130, 176, 260

abominable - 32, 51, 71, 122, 161

abominably - 48, 133, 269, 299

abominate - 263, 296

abound - 101

above - 11, 32, 153, 179, 195, 202, 210, 212, 213, 214, 218, 220, 232, 237, 256, 257, 262, 278, 284

abroad - 194, 196, 233, 288

abrupt - 203

abruptly - 41, 155

abruptness - 198

absence - 54, 56, 64, 77, 78, 90, 99, 100, 106, 110, 111, 127, 150, 172, 194, 195, 197, 205, 207, 224, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 250, 251, 252, 253, 254, 255, 256, 257, 258, 259, 260, 261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 271, 272, 273, 274, 275, 276, 277, 278, 279, 280, 281, 282, 283, 284, 285, 286, 287, 288, 289, 290, 291, 292, 293, 294, 295, 296, 297, 298, 299, 300, 301, 302, 303, 304, 305, 306, 307, 308, 309, 310, 311, 312, 313, 314, 315, 316, 317, 318, 319, 320, 321, 322, 323, 324, 325, 326, 327, 328, 329, 330, 331, 332, 333, 334, 335, 336, 337, 338, 339, 340, 341, 342, 343, 344, 345, 346, 347, 348, 349, 350, 351, 352, 353, 354, 355, 356, 357, 358, 359, 360, 361, 362, 363, 364, 365, 366, 367, 368, 369, 370, 371, 372, 373, 374, 375, 376, 377, 378, 379, 380, 381, 382, 383, 384, 385, 386, 387, 388, 389, 390, 391, 392, 393, 394, 395, 396, 397, 398, 399, 400, 401, 402, 403, 404, 405, 406, 407, 408, 409, 410, 411, 412, 413, 414, 415, 416, 417, 418, 419, 420, 421, 422, 423, 424, 425, 426, 427, 428, 429, 430, 431, 432, 433, 434, 435, 436, 437, 438, 439, 440, 441, 442, 443, 444, 445, 446, 447, 448, 449, 450, 451, 452, 453, 454, 455, 456, 457, 458, 459, 460, 461, 462, 463, 464, 465, 466, 467, 468, 469, 470, 471, 472, 473, 474, 475, 476, 477, 478, 479, 480, 481, 482, 483, 484, 485, 486, 487, 488, 489, 490, 491, 492, 493, 494, 495, 496, 497, 498, 499, 500, 501, 502, 503, 504, 505, 506, 507, 508, 509, 510, 511, 512, 513, 514, 515, 516, 517, 518, 519, 520, 521, 522, 523, 524, 525, 526, 527, 528, 529, 530, 531, 532, 533, 534, 535, 536, 537, 538, 539, 540, 541, 542, 543, 544, 545, 546, 547, 548, 549, 550, 551, 552, 553, 554, 555, 556, 557, 558, 559, 560, 561, 562, 563, 564, 565, 566, 567, 568, 569, 570, 571, 572, 573, 574, 575, 576, 577, 578, 579, 580, 581, 582, 583, 584, 585, 586, 587, 588, 589, 590, 591, 592, 593, 594, 595, 596, 597, 598, 599, 600, 601, 602, 603, 604, 605, 606, 607, 608, 609, 610, 611, 612, 613, 614, 615, 616, 617, 618, 619, 620, 621, 622, 623, 624, 625, 626, 627, 628, 629, 630, 631, 632, 633, 634, 635, 636, 637, 638, 639, 640, 641, 642, 643, 644, 645, 646, 647, 648, 649, 650, 651, 652, 653, 654, 655, 656, 657, 658, 659, 660, 661, 662, 663, 664, 665, 666, 667, 668, 669, 670, 671, 672, 673, 674, 675, 676, 677, 678, 679, 680, 681, 682, 683, 684, 685, 686, 687, 688, 689, 690, 691, 692, 693, 694, 695, 696, 697, 698, 699, 700, 701, 702, 703, 704, 705, 706, 707, 708, 709, 710, 711, 712, 713, 714, 715, 716, 717, 718, 719, 720, 721, 722, 723, 724, 725, 726, 727, 728, 729, 730, 731, 732, 733, 734, 735, 736, 737, 738, 739, 740, 741, 742, 743, 744, 745, 746, 747, 748, 749, 750, 751, 752, 753, 754, 755, 756, 757, 758, 759, 760, 761, 762, 763, 764, 765, 766, 767, 768, 769, 770, 771, 772, 773, 774, 775, 776, 777, 778, 779, 780, 781, 782, 783, 784, 785, 786, 787, 788, 789, 790, 791, 792, 793, 794, 795, 796, 797, 798, 799, 800, 801, 802, 803, 804, 805, 806, 807, 808, 809, 810, 811, 812, 813, 814, 815, 816, 817, 818, 819, 820, 821, 822, 823, 824, 825, 826, 827, 828, 829, 830, 831, 832, 833, 834, 835, 836, 837, 838, 839, 840, 841, 842, 843, 844, 845, 846, 847, 848, 849, 850, 851, 852, 853, 854, 855, 856, 857, 858, 859, 860, 861, 862, 863, 864, 865, 866, 867, 868, 869, 870, 871, 872, 873, 874, 875, 876, 877, 878, 879, 880, 881, 882, 883, 884, 885, 886, 887, 888, 889, 890, 891, 892, 893, 894, 895, 896, 897, 898, 899, 900, 901, 902, 903, 904, 905, 906, 907, 908, 909, 910, 911, 912, 913, 914, 915, 916, 917, 918, 919, 920, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000

absent - 31, 199, 225, 229

absolute - 78, 227, 253, 308

absolutely - 17, 25, 32, 92, 94, 125, 147, 166, 167, 171, 190, 203, 242, 260, 269, 299, 304

absurd - 61, 163, 171, 296, 302

absurdities - 127, 217

absurdity - 189

abundant - 227

abundantly - 67, 85, 125

abuse - 6, 166

abused - 179, 197

abusing - 31, 299

abusive - 184, 316

accede - 166