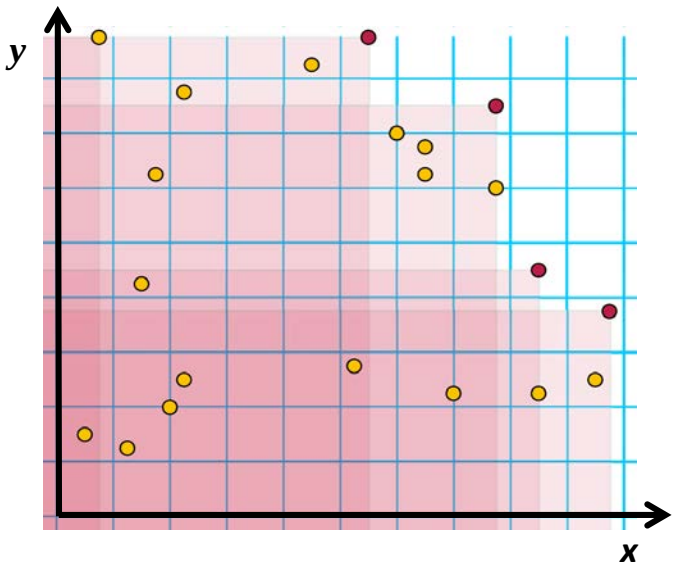
Prog. and Data Structures Lab (Section -1) Spring 2021

Assignment Set 5

---

1. **The Dominance Relation.** You are given a set of points in the 2-dimensional space. The coordinates of the points are given as (*x, y*) pairs.
   - A point $P_1 = (x_1 , y_1)$ is *dominated* by a point $P_2 = (x_2 , y_2)$ if $y_2 > y_1$ and $x_2 > x_1$
   - A point is *non-dominated* in a set of points, if no point dominates it in that set

   As an example, the red points in the figure are non-dominated points. The orange points are dominated by one or more non-dominated points. Finding such non-dominated points is very important in multi-criteria optimization.



Write a program with the following:

(a) Define a structure, `struct point`, with two floating point attributes, `xcord` and `ycord`.

(b) Read a set of points from an input file, `inp.dat`, into an array of such structures. The format of the input is as follows. The first line will have the number of points, *N*. Each of the next *N* lines will have the x-coordinate and y-coordinate of a point separated by a space. For example, for the set of five points: {(2.5, 7.2), (4.7, 5), (9, 5.9), (5.6, 9.5), (12.6, 2.3)}, the input file is as follows:

```
5
2.5 7.2
4.7 5
9 5.9
5.6 9.5
12.6        2.3
```

(c) Write a function, `dominance()` (with suitable arguments) to find and return the set of non-dominated points in the set of points read in part-2. In this function, add comments at the beginning to outline the method used to find the non-dominated points.

(d) The *level of dominance* of a point is the number of non-dominated points that dominate it. The figure shows the regions at different levels of dominance using different shades. Write a function, `level_of_dominance()` (with suitable arguments) to print the list of points along with their level of dominance. For example, if a point, P = (2, 3), is dominated by 5 non-dominated points, then corresponding to this point, the function should print:

```
(2,3) is dominated by 5 non-dominated points
```

(e) Write a main program which does the following:

i. Reads the points from the input file.

ii. Calls the function `dominance()` and from the returned set, prints the set of non-dominated points.

iii. Calls the function `level_of_dominance()` which results in printing each point and its level of dominance.

[Here is a code snippet that reads N and then reads and prints N floating point numbers:

```
FILE *fopen(), *fp; // fp is a file pointer
int N, i;
float x;
fp = fopen("input.dat", "r"); // This opens the file for reading
fscanf(fp, "%d", &N); // Read int N from the file
for (i=0; i<N; i++) {
      fscanf(fp, "%f", &x); // Read float x from the file
      printf("%f\n", x);
}
```

]

2. **Dealing with Sentences.** Reading a sentence and extracting syntactic and semantic information from it is at the heart of natural language processing. This is a simple assignment on sentence parsing. Write a C program which does the following:

(a) It reads a sentence having multiple words. [Hint: Look up `scanf()` with `"%[^\n]"`]

(b) It reports the number of words in the sentence, the shortest and longest words in the sentence, the first and last word in alphabetical order. You are encouraged to use the string library.

(c) A *sentence anagram* of a sentence is a rearrangement of the words in the sentence (ignore punctuations). For example, "Flying across the meadows we saw the sheep" is a sentence anagram of "Across the meadows we saw the flying sheep". Your C program should read another sentence and report whether it is a sentence anagram of the first one.