

# Data Structures and Algorithms

## Exercise 2

Markus Innerebner, Romans Kasperovics  
dsa(a)inf.unibz.it

Free University of Bozen - Bolzano, Italy

Thursday, March 5rd, 2009

## Solutions 01

### Task 2

Implement the Insertion Sort algorithm to sort an array of  $n$  integers and measure the running time.

```
void insertionSort() {
    for(j=1; j<n; j++) {
        key=A[j];
        i=j-1;
        while(i>=0 && A[i]>key) {
            A[i+1]=A[i];
            i--;
        }
        A[i+1]=key;
    }
}
```

Measurement: Run time

$n$	time [sec]
100	0.00
1 000	0.00
10 000	0.23
100 000	15.63

## Solutions 01

### Task 1

Write a C program that outputs a two-dimensional array  $B[n][n]$  in which the entry  $B[i][j]$  for  $i \leq j$  contains the sum of the array entries  $A[i]$  through  $A[j]$ .

```
#include <stdio.h>
#define n 6
int A[n] = {2,7,1,5,3,0}; int B[n][n]; // output
int main(){
    int i,j;
    for (i=0; i<n; i++) {
        for (j=0; j<n; j++) {
            if (i > j) B[i][j] = 0;
            //if (i == j) B[i][j] = A[i];
            if (i <= j) B[i][j] = B[i][j-1] + A[j];
            printf("%2i ",B[i][j]);
        }
        printf("\n");
    }
}
```

$A[n]$

2	7	1	5	3	0
---	---	---	---	---	---

$B[n][n]$

2	9	10	15	18	18
0	7	8	13	16	16
0	0	1	6	9	9
0	0	0	5	8	8
0	0	0	0	3	3
0	0	0	0	0	0

## Solutions 01

### Task 3: Sierpinski Triangle 1/3

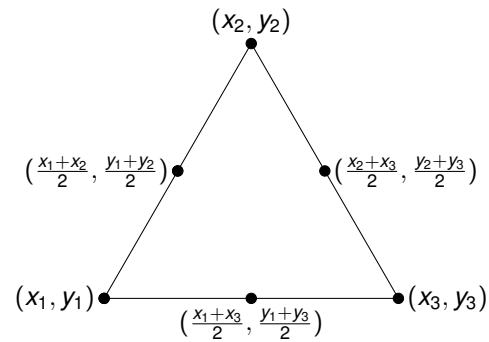
Implement a recursive algorithm that prints the drawing instructions for  $i$ -th iteration with glut.

```
void drawTriangle(float x1, float y1, float x2, float y2, float x3, float y3) {
    glBegin(GL_TRIANGLES); glColor3f(0,0,0); // color of triangle
    glVertex3f(x1,y1,0); // left
    glVertex3f(x2,y2,0); // upper
    glVertex3f(x3,y3,0); // right
    glEnd(); glFlush();
}

void ST(float x1, float y1, float x2, float y2, float x3, float y3, int n){
    if(n==0) {drawTriangle(x1,y1,x2,y2,x3,y3);}
    else {
        float x12=(x1+x2)/2, y12=(y1+y2)/2, x13=(x1+x3)/2, y13=(y1+y3)/2,
            x23=(x2+x3)/2, y23=(y2+y3)/2;
        ST(x1, y1, x12, y12, x13, y13, n-1);
        ST(x12, y12, x2, y2, x23, y23, n-1);
        ST(x13, y13, x23, y23, x3, y3, n-1);
    }
}
```

## Solutions 01

### Task 3: Sierpinski Triangle 2/3



## Solutions 01

### Task 3: Sierpinski Triangle 3/3

```
// Some magic to draw with GL inside a window
void DrawGLScene() {
    glClearColor(1,1,1,0);
    glClear(GL_COLOR_BUFFER_BIT);
    ST(-0.8,-0.8,0.8,-0.8,0,0.8,d); // invoking Sierpinski's triangle
}
// The main method with some magic to create a window
int main(int argc, char **argv) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
    glutCreateWindow("Sierpinski Triangle");
    sscanf(argv[1], "%d", &d);
    glutDisplayFunc(DrawGLScene);
    glutMainLoop();
}
```