Programming for computerteknologi
Hand-in Assignment Exercises

Week 4: Structured data and pointers

Exercise 1)
A function called *area* should be able to calculate and return the *area* of a rectangle as an integer given four different input variables. This will give us the following function:

```
3   int area (int x1, int x2, int y1, int y2) {
4       return (x2 - x1) * (y2 - y1);
5   }
```

The function is called *area* and it needs to return an integer which gives us an integer function called *area*. Then we define the input values, which is the four integer coordinates used to calculate the area of the rectangle.

Exercise 2)
A function called *increment* should be able to take a pointer to an integer and add 1 to the integer value its pints. With this in mind, we get the following function:

```
1   #include <stdio.h>
2
3   void increment (int *x) {
4       *x+=1;
5   }
6
7   int main() {
8       int v = 5;
9       increment(&v);
10      printf("%d", v);
11      return 0;
12  }
```

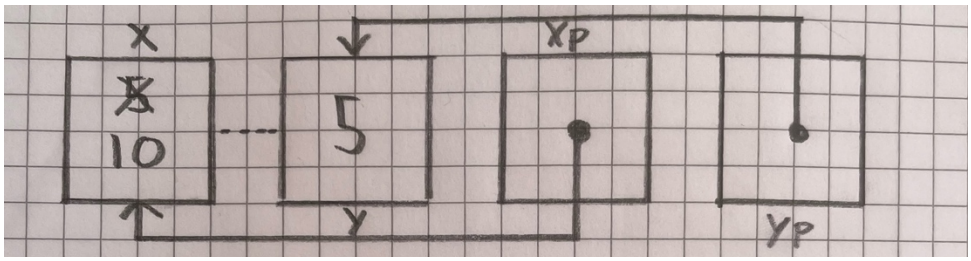
The function is called *increment* and it doesn't have to return anything which gives us a void function called *increment*. Then we reference to the pointer v and adds one to it which gives 6.

Exercise 3)

We have been given a code to analyze. We had to calculate what the values for $x, y, *xp, *yp$ by the end of the code. The values printed out will be the following:

$x = 10, y = 5, *xp = 10, *yp = 5$

I have also drawn a diagram as in the lectures to explain my answer. This diagram is included and pictured below:

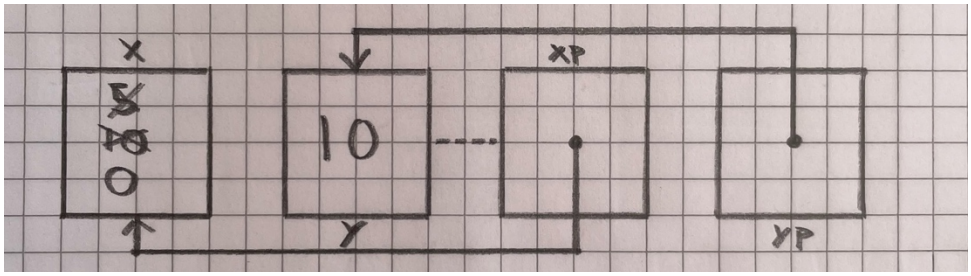


Exercise 4)

We have been given a code to analyze. We had to calculate what the values for $x, y, *xp, *yp$ by the end of the code. The values printed out will be the following:

$x = 0, y = 10, *xp = 0, *yp = 10$

I have also drawn a diagram as in the lectures to explain my answer. This diagram is included and pictured below:

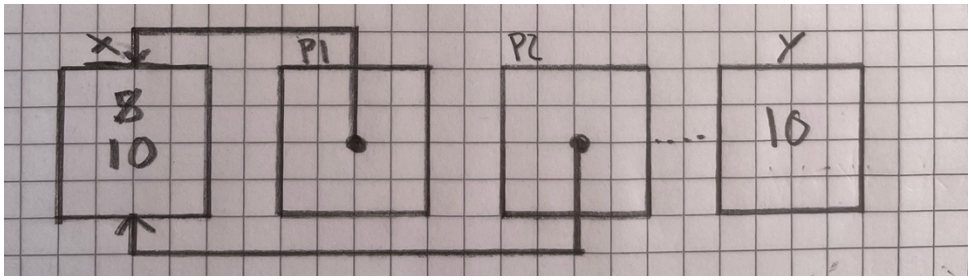


Exercise 5)

We have been given a code to analyze. We had to calculate what the values for $x, y, *xp, *yp$ by the end of the code. The values printed out will be the following:

$x = 10, y = 10, *p1 = 10, *p2 = 10$

I have also drawn a diagram as in the lectures to explain my answer. This diagram is included and pictured below:



Exercise 6)

a)

I have written a code that represents a circle using C struct. I have made a typedef that represents a point that consists of a x-coordinate and an y-coordinate, which together make up the center of the circle. I have next made a new typedef that includes a variable for the circle's radius and a pointer to the circles center coordinates. The structures can be seen below:

```c
#include <stdio.h>
#include <assert.h>

typedef struct {
    int x;
    int y;
} point;

typedef struct {
    int r; /* radius */
    point p; /* pointer coordinates */
} circle;
```

b)

I have made a code that takes inputs such as the circles radius and its x- and y-coordinates from the user and store it in an array. The code can be seen below:

```c
int main() {
    circle c[5]; /* array with number of circles */
    int i; /* counter variable */
    int q; /* placeholder */

    for (i = 0; i < 5; i++) {
        printf("Insert radius, x coordinate and y cordinate to the center of circle %d:\n", q = i + 1);
        scanf("%d%d%d", &c[i].r, &c[i].p.x, &c[i].p.y);
    }
}
```

c)

I have now made a function that takes a pointer to a circle as input and returns either true or false if the circle's radius is greater than zero or not. The function as well as implementation in the main function can be seen below:

```c
int circleIsValid(const circle *c) {
    int i; /* counter variable */
    int q; /* placeholder */

    for (i = 0; i < 5; i++) {
        if (c[i].r>0) {
            q = 1;
        }
        else {
            q = 0;
            return q;
        }
    }
    return q;
}
```

```c
int main() {
    circle c[5]; /* array with number of circles */
    int i; /* counter variable */
    int q; /* placeholder */
    int isValid; /* to check precondition */

    for (i = 0; i < 5; i++) {
        printf("Insert radius, x coordinate and y cordinate to the center of circle %d:\n", q = i + 1);
        scanf("%d%d%d", &c[i].r, &c[i].p.x, &c[i].p.y);
    }

    q = circleIsValid(c);
    if (q == 1) {
        printf("true\n");
    }
    else {
        printf("false\n");
        abort();
    }
    return 0;
}
```

d)

Lastly, I have made a function that takes a pointer to a geometric point and adds some newly given coordinate values. This function as well as test cases can be seen below:

```c
void translate(circle *c, const point *p) {
    int i; /* counter variable */
    int q; /* placeholder */
    int a, b; /* new input coordinates (a,b) */

    for (i = 0; i < 5; i++) {
        printf("Insert new coordinates for circle number %d:\n", q = i + 1);
        scanf("%d%d", &a, &b);
        a += c[i].p.x;
        b += c[i].p.y;
        printf("When adding to the geometric point (%d,%d) we get the following coordinates:\n(%d,%d)\n", c[i].p.x, c[i].p.y, a, b
    }
}
```

```
Insert radius, x coordinate and y cordinate to the center of circle 1:
5 4 3
Insert radius, x coordinate and y cordinate to the center of circle 2:
7 6 4
Insert radius, x coordinate and y cordinate to the center of circle 3:
9 7 9
Insert radius, x coordinate and y cordinate to the center of circle 4:
1 5 3
Insert radius, x coordinate and y cordinate to the center of circle 5:
8 9 9
true
```

```
Insert new coordinates for circle number 1:
1 5
When adding to the geometric point (4,3) we get the following coordinates:
(5,8)
Insert new coordinates for circle number 2:
4 2
When adding to the geometric point (6,4) we get the following coordinates:
(10,6)
Insert new coordinates for circle number 3:
4 5
When adding to the geometric point (7,9) we get the following coordinates:
(11,14)
Insert new coordinates for circle number 4:
1 2
When adding to the geometric point (5,3) we get the following coordinates:
(6,5)
Insert new coordinates for circle number 5:
1 1
When adding to the geometric point (9,9) we get the following coordinates:
(10,10)
```

Exercise 7)

a)

I have made a function that calculates the difference between the values of numbers in an array. If the difference is less than the amount of numbers in the sequence and when the absolute difference between the values comes in order (for instance 1, 2, 3, 4 or 1, 4, 3, 2, 5). If that's the case, then the array is Jolly. The code is pictured below:

```
7   int isJollyJumper(const int seq[], int size) {
8       int i;
9       int diff;
10      bool diffs_found[size];
11      int jolly;
12
13      for (i = 0; i < size; i++) {
14          diffs_found[i] = false;
15      }
16
17      for (i = 1; i <= size; i++) {
18          diff = seq[i] - seq[i - 1];
19          if (diff < 0) {
20              diff = diff * -1;
21          }
22          if (diffs_found[diff - 1] == true || diff > size) {
23
24              jolly = 0;
25              return jolly;
26          }
27          else {
28              diffs_found[diff - 1] = true;
29          }
```

b)
I have made a function that takes input from the user and stores it in an array. Then the newly created function will be called and checks if the given array is jolly or not. The function is pictured below:

```
38  int main() {
39      int i; /* counter variable */
40      int n;
41      int seq[n];
42      int jolly;
43
44      printf("How long do you want the sequence to be?\n");
45      scanf("%d", &n);
46
47      printf("Insert sequence that's %d numbers long:\n", n);
48      for (i = 0; i < n; i++) {
49          scanf("%d", &seq[i]);
50      }
51
52      jolly = isJollyJumper(seq, n - 1);
53
54      if (jolly == 0) {
55          printf("Not jolly");
56      }
57      else if (jolly == 1) {
58          printf("Jolly");
59      }
60
61      return 0;
```