1) (Text answer) A function area calculates and returns the area of a rectangle as an integer. The input rectangle is given as four integer coordinates: *x1, x2, y1* and *y2*. Complete the function signature below.

```
1
2
3 _________  __________ ( ____________________________________ ) {
4
5   return (x2 - x1) * (y2 - y1);
6 }
```

Denne funktion skal være en int, da opgaven viser den skal return et tal fra en udregning. Funktionen bruger x1, x2, y1 og y2 hvor værdierne bliver hentet fra main. Hvis man skal skrive funktionen skal den se således ud.

void areaCalculation(int x1, int x2, int y1, int y2){
Return (x2- x1) * (y2 – y1);
}

2) (Text answer) The function increment takes a pointer to an integer and adds 1 to the integer value to which its pints. The function does not return any value. Complete the function signature and function body below, so that the main function prints 6 when executed.

```
1
2
3 __________  ____________________ ( _______________ ) {
4
5
6   ____________________;
7 }
8
9 int main () {
0   int v = 5;
1   increment(&v);
2   printf("%d", v);
3   return 0;
4 }
```

Opgaven siger at functionen ikke retuner en værdi, så derfor er det en void funktion. I main bliver funktionen kaldt ved increment og bruger værdien fra v.

Så indtil videre ser funktionen sådan her ud.

Void increment(int *v){

}

*v bliver brugt fordi opgaven går ud på funktionen bruger en pointer.

Hvis man lægger et tal til eller giver pointeren en ny værdi, får variabel som pointeren er forbundet med den samme værdi.

Så funktionen kommer til at se således ud

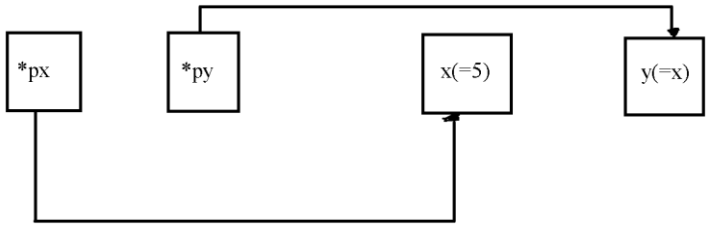
```
Void increment(int *v){
        *v += 1;
}
```

3) (Text answer) Consider the following code. At the end of the function, what are the values x, y, *xp, *yp? Your submission must include your diagram.

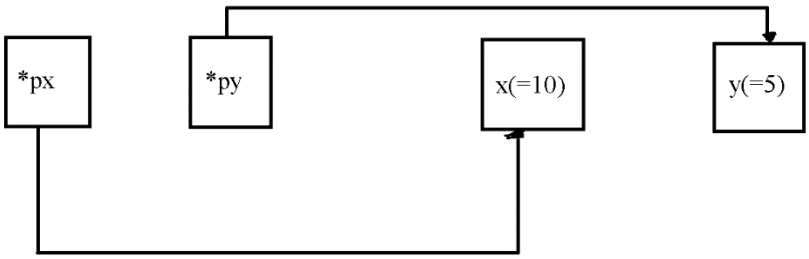
```
int main(void)
{
  int x;
  int y;

  int *xp;
  int *yp;

  x = 5;
  y = x;

  xp = &x;
  yp = &y;

  x = 10;

  /* What are values of: x,y,*xp,*yp */

  printf("x=%d, y=%d, *xp=%d, *yp=%d\n", x,y,*xp,*yp);

  return 0;
}
```

Programmet printer x = 10, y = 5, xp = 10, yp = 5.

Fordi som i sidste opgave ændrer man på enten pointeren eller værdien som bliver pointet på, så ændre de begge sig til den givende værdi, og da y er defineret som x inden den får en ny værdi og da y ikke er en pointer på x så ændres dens værdi sig ikke.



Figur 1Diagram x = 5



Figur 2Diagram x bliver ændret til 10

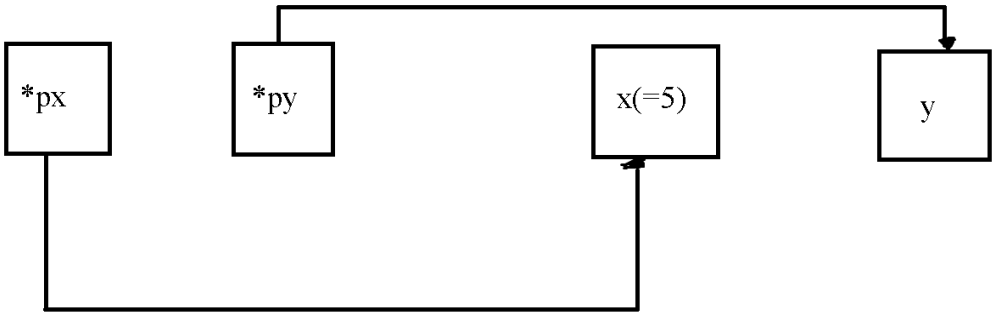
4) (Text answer) Consider the following code. At the end of the function, what are the values x, y, *xp, *yp? Your submission must include your diagram.

```
int main(void)
{
  int x;
  int y;

  int *xp;
  int *yp;

  x = 5;

  xp = &x;

  x = 10;

  y = *xp;

  yp = &y;

  *xp = 0;

  /* What are values of: x,y,*xp,*yp */

  printf("x=%d, y=%d, *xp=%d, *yp=%d\n", x,y,*xp,*yp);

  return 0;
```

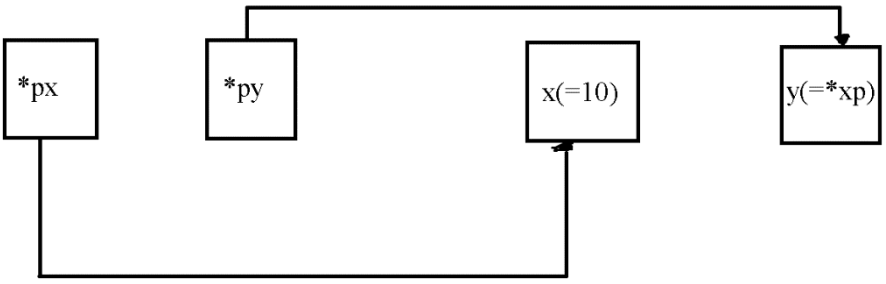
Princippet er lige som i sidste opgave, så vil sprænge over forklaring og vil kun forklar de ting som er ændret ved denne kode. Nu er y defineret som pointeren til x og ikke x og til sidst er pointeren til x sat til 0.

I sidste opgave blev der beskrevet at når en værdi eller pointeren ændrer værdi så ændre begge til den given værdi, så x og xp ændrer med at være 0.

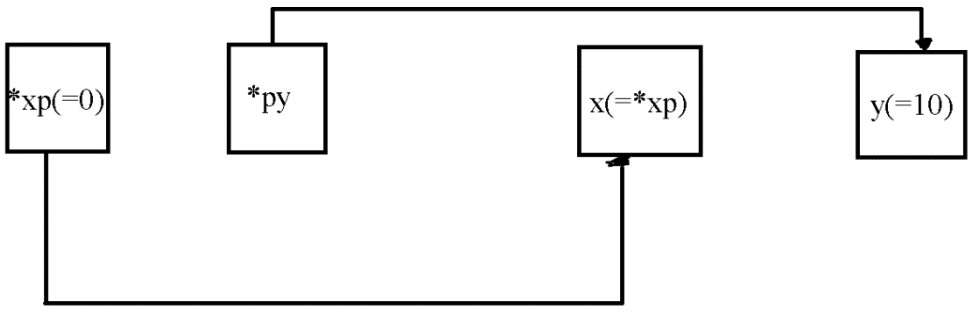
Y er defineret ved *xp, altså pointeren til x. På dette tidspunkt i koden har *xp værdien 10 og derfor får y og *yp værdien 10. Da y ikke er en pointer med *xp eller x så beholder både y og *yp altså værdien 10 igennem hele programmet.
Så programmet printer x = 0, y = 10, *xp = 0, *yp = 10.



Figur 3Daigram x=5, y ikke defineret



*Figur 4Diagram x = 10, y = *xp*

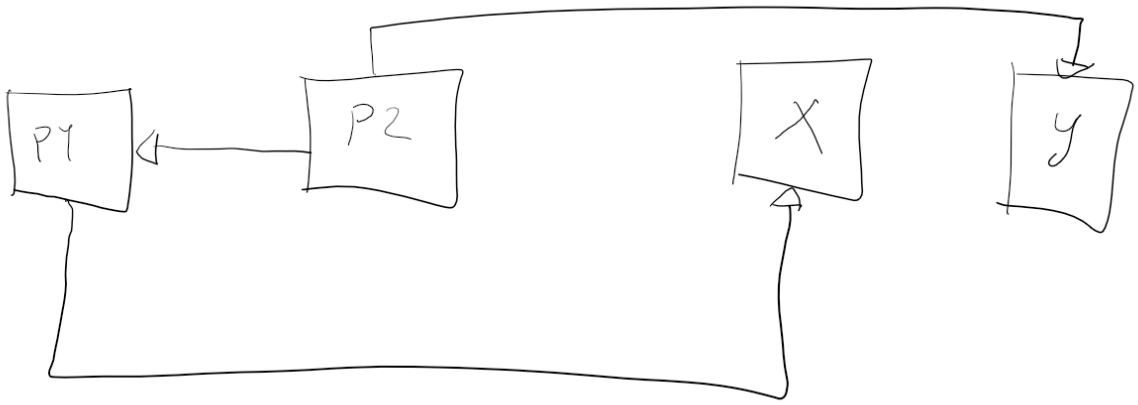


*Figur 5Diagram *xp = 0, y = 10*

5) (Text answer) Consider the following code. At the end of the function, what are the values x, y, *p1, *p2? Your submission must include your diagram.

```
int main(void)
{
  int x;
  int y;

  int *p1;
  int *p2;

  x = 5;
  y = 10;

  p1 = &x;
  p2 = p1;

  *p2 = y;

  p1 = &x;

  /* What are values of: x,y,*xp,*yp */
```

P1 starter med at have værdien 5 da den peger på x, derefter får p2 værdien 5 da den peger på p1. Nu peger p2 så på y og får værdien 10, og da p2 også peger på p1 får p1 værdien 10, p1 peger på x og nu får x værdien 10.



Figur 6Diagram for opgave5

6) (Code answer) In the lecture we discussed how to represent a geometric point using a C struct. Let's now consider a geometric circle: a circle consists of three integers: x coordinate of the center point, y coordinate of the center point and a radius.
    a. Write a C strut that represent a circle using C strut with an integer representing the radius (named r) and a point (named p and using the struct down in class)
    b. Create an array of five circles c[5] such that circle c_i has center point (i, i) and radius i
    c. Create a function *CircleIsValid* that takes a pointer to a circle as input and returns true if the radius is positive (r > 0) and false otherwise. The function should have the following signature *Int CircleIsValid(const circle *c)*
    d. Create a function translate that takes a pointer to a circle c and a pointer a geometric point p, and adds the coordinate's value of p to the center point coordinate values of c. The function should have the following signature *void translate(circle *c, const point *p)*

7) (Code answer) (PC-2.8.1)
    a. A sequence of n > 0 integers is called a jolly jumper if the values of the difference between successive elements take on all possible values 1 through n – 1 (The order of the difference does not matter). The definition implies that any sequence of a single integer is a jolly jumper. The function should have the following signature *int isJollyJumper(const int seq[ ], int size)*
    b. Write a test program that reads the size and sequence. And uses the function to print out if the sequence is a Jolly Jumper or not:
        i. **Input:** A line of input contains an integer n < 100 followed by n integers representing the sequence.
        ii. **Output:** For the line of input generate a line output saying "Jolly" or "Not Jolly"