Programmering for computerteknologi
Hand-in Assignment Exercises

Week 5: Structured data and pointers

Please make sure to submit your solutions **by next Monday**.

In the beginning of each question, it is described what kind of answer that you are expected to submit. If *Text and code answer* is stated, then you need to submit BOTH some argumentation/description and some code; if just (*Text answer*) or (*Code answer*) then just some argumentation/description OR code. The final answer to the answers requiring text should be **one pdf document** with one answer for each text question (or text and code question). When you hand-in, add a link to your GitHub reposetory in the beginning of your pdf file. Make sure that you have committed your code solutions to that reposetory.

Note: the **Challenge** exercises are *optional*, the others mandatory (i.e. you **have** to hand them in).

Exercises

(1) (Text answer) (Old exam question) 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`, `y2`. Complete the function signature below.

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

(2) (Text answer) (Old exam question) The function `increment` takes a pointer to an integer and adds `1` to the integer value to which it 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 () {
10   int v = 5;
11   increment(&v);
12   printf("%d", v);
13   return 0;
14 }
```

(3) (Text answer) Consider the following code. At the end of the function, what are the values for $x, y, *xp, *yp$? Using pen and paper, draw a diagram (like in the lectures) to explain your answer. Your submission must include your diagram. The following diagram formats are allowed: PDF, JPG and PNG.

```
#include <stdio.h>

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;
}
```

(4) (Text answer) Consider the following code. At the end of the function, what are the values for $x, y, *xp, *yp$? Using pen and paper, draw a diagram (like in the lectures) to explain your answer. Remember to include your diagram (in PDF, JPG or PNG format) in your submission.

```
#include <stdio.h>

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;
}
```

(5) (Text answer) Once again, consider the following code. At the end of the function, what are the values for $x, y, *xp, *yp$? Using pen and paper, draw a diagram (like in the lectures) to explain your answer. Remember to include your diagram (in PDF, JPG or PNG format) in your submission.

```
#include <stdio.h>

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 */
```

```
  printf("x=%d, y=%d, *p1=%d, *p2=%d\n", x,y,*p1,*p2);

  return 0;
}
```

(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 centre point, y coordinate of the centre point, and a *radius*.
   (a) Write a C struct that represents a *circle* using a C struct with an integer representing the radius (named `r`) and a *point* (named `p` and using the struct shown in class).
   (b) Create an array of five circles $c[5]$ such that circle c_i has centre 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 of the circle 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 to a geometric *point* p (like in the lecture), and adds the coordinate values of p to the centre point coordinate values of c, i.e. it translates the circle by a vector represented by the point p. For example, if c is initially centred at $(5, 10)$ and we pass, as input, a point p with coordinate values $(1, -1)$ then the centre point of c becomes $(6, 9)$ after the *translate* function. 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 $\mathsf{n} > 0$ integers is called a *jolly jumper* if the absolute values of the differences between successive elements take on all possible values 1 through $\mathsf{n} - 1$. For instance, 1 4 2 3 is a jolly jumper, because the absolute differences are 3, 2, and 1, respectively. As another example, 11 7 4 2 1 6 is a jolly jumper, because the absolute differences are 4, 3, 2, 1, 5 (the order of the differences does not matter). The definition implies that any sequence of a single integer is a jolly jumper. Write a function to determine whether a sequence is a jolly jumper. The function should have the following signature: `int isJollyJumber(const int seq[], int size)` (Hint: use a boolean array, e.g. `bool diffs_found[n]` to keep track of the differences found so far between consecutive numbers. So that `diffs_found[2]` being `true` implies that the absolute difference `2` has already been found.
   (b) Write a test program that reads the size and sequence, and uses the function to print out if the sequence is a JollyJumper or not.:
      Input A line of input contains an integer $\mathsf{n} < 100$ followed by n integers representing the sequence.
      Output For the line of input generate a line of output saying "Jolly" or "Not jolly".
      Example

```
4
1 4 2 3
Jolly

5
1 4 2 -1 6
Not jolly
```

(8) **Challenge:** (PC-2.8.8) The game of Yahtzee involves five dice, which are thrown in 13 rounds. A score card contains 13 categories. Each round may be scored in a category of the player's choosing, but each category may be scored only once in the game. The 13 categories are scored as follows:

- **ones** – sum of all ones thrown
- **twos** – sum of all twos thrown
- **threes** – sum of all threes thrown
- **fours** – sum of all fours thrown
- **fives** – sum of all fives thrown
- **sixes** – sum of all sixes thrown
- **chance** – sum of all dice
- **three of a kind** – sum of all dice, provided at least three have same value
- **four of a kind** – sum of all dice, provided at least four have same value
- **five of a kind** – 50 points, provided all five dice have same value
- **short straight** – 25 points, provided four of the dice form a sequence (that is, 1,2,3,4 or 2,3,4,5 or 3,4,5,6)
- **long straight** – 35 points, provided all dice form a sequence (1,2,3,4,5 or 2,3,4,5,6)
- **full house** – 40 points, provided three of the dice are equal and the other two dice are also equal. (for example, 2,2,5,5,5)

Each of the last six categories may be scored as 0 if the criteria are not met. The score for the game is the sum of all 13 categories plus a bonus of 35 points if the sum of the first six categories is 63 or greater. Your job is to compute the best possible score for a round.

Input Each line of input contains five integers between 1 and 6, indicating the values of the five dice thrown in each round. There are 13 such lines for each round.

Output Your output should consist of a single line for the round containing 15 numbers: the score in each category (in the order given), the bonus score (0 or 35), and the total score. If there is more than categorisation that yields the same total score, any one will do.

Example

```
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
```

```
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5 0 15 0 0 0 25 35 0 0 90

1 1 1 1 1
6 6 6 6 6
6 6 6 1 1
1 1 1 2 2
1 1 1 2 3
1 2 3 4 5
1 2 3 4 6
6 1 2 6 6
1 4 5 5 5
5 5 5 5 6
4 4 4 5 6
3 1 3 6 3
2 2 2 4 6
3 6 9 12 15 30 21 20 26 50 25 35 40 35 327
```