Assignment 2

This assignment is to be done individually.

Important Note: The university policy on academic dishonesty (cheating) will be taken very seriously in this course. You may not provide or use any solution, in whole or in part, to or by another student.

You are encouraged to discuss the concepts involved in the questions with other students. If you are in doubt as to what constitutes acceptable discussion, please ask! Further, please take advantage of office hours offered by the instructor and the TA if you are having difficulties with this assignment.

DO NOT:

- Give/receive code or proofs to/from other students
- Use search engines to find solutions for the assignment

DO:

- Meet with other students to discuss assignment (it is best not to take any notes during such meetings, and to re-work assignment on your own)
- Use online resources (e.g. Wikipedia) to understand the concepts needed to solve the assignment

Submission Instructions:

- You may type or write your answer as long as it is readable.
- Submit two files on CourSys
  1. `report.pdf`, which contains a write-up of your solutions to the assignment
  2. `replication.c`, which contains the code you wrote in Problem 3.

Assignment 2

Problem 1

For the following statements indicate which one is True/False. For the False statement(s), write the correct from. (8 marks)

a) The snippet bellow is a part of a larger code in which `pA1` refers to a integer array of size 10. All pointers (`pA1, pA2, pA3`) after line 8 will point to the same variable in the memory.

```
1  #include <stdio.h>
2
3  int main() {
4      ...
5      int a[10];
6      a[0] = *pA1;
7      int *pA2 = &a[1]-1;
8      pA3 = a;
9      ...
10     return 0;
11 }
```

b) In the previous code in part a), the values of `*pA1+1`, `*pA2+1`, `*pA3+1` are the same but the values of `*(pA1+1)`, `*(pA2+1)`, `*(pA3+1)` may be different.

c) If `pA` is a pointer to an array `a`, the following line of code causes a segmentation fault. That is, the code may compile but running the code could result in run-time error.

```
1  pA = a[1];
```

d) The minimum required change for the following snippet to show all of the values in array `a` on separate lines is adding or deleting one character.

```
1  #include <stdio.h>
2
3  int main() {
4      int a[10];
5      for (int i=a; i<=a+9;i++)
6      printf("%d\n",i);
7      return 0;
8  }
```

e)

$$O\Big(\big(\frac{1}{n}(\log_2 n)^2 + \frac{1}{\sqrt{n}}\big)\big(\sqrt{n}\log_3(\log_2 n) + \sqrt{n}\log_2 n\big)\Big) = O\Big(\frac{(\log n)^3}{\sqrt{n}}\Big)$$

Problem 2

Give the best Big O estimate of the worst run time for the following code snippets. (9 marks)

a)

```
#include <stdio.h>

void main() {
    int i, j;
    int n;
    int K = 1000;

    for (i = K; i <= n; i++) {
        j = 3*i;
        while (j <= n) {
            j++;
            for (i=1; i<=K ; i++){
                printf("programming is fun\n");
            }
        }

    }
}
```

b)

```
#include <stdio.h>

void main() {
    int i, j;
    int n;

    for (i = 1; i <= n; i++) {
        if (i %2 == 0){
            for (j=i; j<n-i ; j++){
               printf("programming is fun\n");
            }
        }
        else{
            for (j=n-i; j<n ; j++){
               printf("programming is fun\n");
            }
        }
    }
```

```
19 }
```

c)

```
1  #include <stdio.h>
2  #include <math.h>
3
4  void func(n){
5      for (int k=1; k<n;k++){
6          printf("programming is fun\n");
7          func(round(n/2));
8      }
9  }
10
11 void main() {
12     int i;
13     int n;
14
15     for (i = 1; i <= n; i++) {
16         func(n);
17
18     }
19 }
```

Problem 3

Write a code (`replication.c`) that prints the longest sub-string from the input that occurs more than once, and that does not overlap with other occurrences of the same substring. Your algorithm should be independent from the input size. Give the best Big-O estimate of your algorithm in the worst case. (13 marks)

(Hint: See examples below. Note that the longest obtained pattern should be from at least two **separable** chunks of the input string. Remember to include `string.h`, which contains the function `strlen`.)

Example 1:

input: `abcab`

output: `ab`

Example 2:

input: `abababababa`

output: `abab` or `baba`