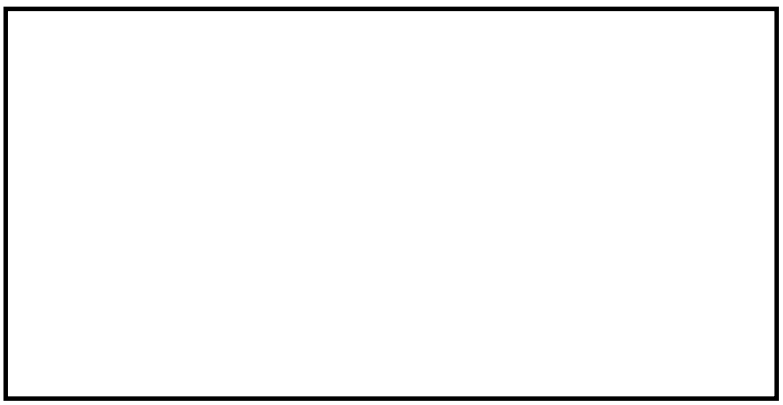
2016.1 Final Exam : Basic Programming

supervisor signature	

Student Id. : ( ) , Name : ( )

1.(20 points) **Fill out the empty box in the following code** to write a program that takes two strings as input and prints out to the screen the longest common prefix (LCP) of the two strings. Assume that the size of a string is less than 100.
(Examples : the longest common prefix of "global" and "glossary" is "glo", of "department" and "depart" is "depart", and of "glove" and "dove" is the empty string "")

```
#include <stdio.h>
void getLCP(char* s1, char* s2, char* LCP)
{
```



```
}
```

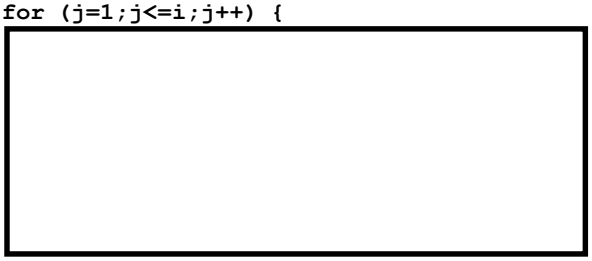
```
int main()
{
  char s1[100],s2[000],LCP_str[100];
  scanf("%s %s",s1,s2);
  getLCP(s1,s2,LCP_str);
  printf("%s\n",LCP_str);
  return 0;
}
```

```
Input Example :
global glossary

Output Example (Execution Result) :
glo
```

2. (20 points) Following source code is a C implementation of bubble sorting method that sorts given array elements in an increasing order. **Fill out the empty box in the code.** Note the array A consists of N integer elements: A[0], A[1], ..., A[N-1]..

```
void bubbleSort(int A[], int N)
{
  int temp, i, j;
  for (i=N-1;i>=1;i--) {
    for (j=1;j<=i;j++) {
```



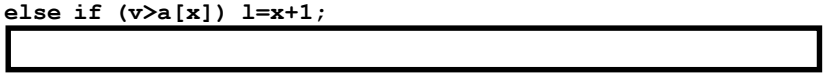
```
    }
  }
}
```

```
Example showing how bubble sorting processes data
First Pass :
        ( 5 1 4 2 8 ) -> ( 1 5 4 2 8 )
        ( 1 5 4 2 8 ) -> ( 1 4 5 2 8 )
        ( 1 4 5 2 8 ) -> ( 1 4 2 5 8 )
        ( 1 4 2 5 8 ) -> ( 1 4 2 5 8 )
Second Pass :
        ( 1 4 2 5 8 ) -> ( 1 4 2 5 8 )
        ( 1 4 2 5 8 ) -> ( 1 2 4 5 8 )
        ( 1 2 4 5 8 ) -> ( 1 2 4 5 8 )
Third Pass :
        ( 1 2 4 5 8 ) -> ( 1 2 4 5 8 )
        ( 1 2 4 5 8 ) -> ( 1 2 4 5 8 )
Forth Pass :
        ( 1 2 4 5 8 ) -> ( 1 2 4 5 8 )
```

3. (16 poitns) Following code is implementaion of binary search algorithm. **Fill out empty boxes with appropriate C code.**

```
int binary_search(int a[], int N, int v)
{ int l,r,x;          //  This function searches for v in array a with size N.
  l=0; r=N-1;         //  If found, it returns the array index. If not found, it returns -1.
  while (r>=l) {
```

```
    if (v<a[x]) r=x-1;
    else if (v>a[x]) l=x+1;
```



```
  }
  return -1;
}
```

4. (20 points) Write a program that simulates the rolling of two dice. The program should use rand() to roll the first die, and should use rand() again to roll the second die. The sum of the two values should then be calculated. [Note: Since each die can show an integer value from 1 to 6, then the sum of the two values will vary from 2 to 12, with 7 being the most frequent sum and 2 and 12 the least frequent sums.] Your program should roll the two dice 36,000 times and print the number of cases among 36,000 and probability for each sum value. The result of your program should look similar to the following output example (exact number may be different).

(1) Fill out empty boxes with appropriate C code.

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
int main() {
        srand(time(NULL));
        int i,dice1,dice2,dice_sum;
        int count[13]={0,0,0,0,0,0,0,0,0,0,0,0,0};
// insert your code here
        return 0;
}
```

Output Example

```
2 : 1026 (0.028500)
3 : 2023 (0.056194)
4 : 2988 (0.083000)
5 : 4086 (0.113500)
6 : 5018 (0.139389)
7 : 5978 (0.166056)
8 : 4928 (0.136889)
9 : 3992 (0.110889)
10 : 3096 (0.086000)
11 : 1907 (0.052972)
12 : 958 (0.026611)
```

(2) What does the fifth line code "`srand(time(NULL));`" do? Why is it necessary?
Explain. (_____)

5. (10 points) Fill out blanks with appropriate English words.

(1) Function call stack supports the function call/return mechanism. Each time a function calls another function, a stack frame (also known as an activation record) is pushed onto the stack. The activation record the return address that the called function needs to return to the calling function. The activation record stores and maintains (a.), (b.), and (c.) when a function is called..

(2) (d.) occurs if the call stack pointer exceeds the stack bound, meaning it occurs when more function calls occur than can have their activation records stored on the function call stack (due to memory limitations)

6. (14 points) Assume **`rand()`** function returns an integer number between 0 and RAND_MAX.

(1) Write a line of C code that assigns a random integer number (**`int`** type) between integer numbers **`a`** and integer **`b`** to **`x`**.
(random integer **`x∈[a,b]`**, where **`x, a, b`** are integer (**`int`** type) numbers.)

Insert your C code in empty box: **`x=`**

(2) Write a line of C code that assigns a random real number (**`float`** type) value between real numbers 0.0 and 1.0 to **`x`**.
(random number **`x∈[0.0,1.0]`**, where **`x is`** a real (**`float`** type) number.)

Insert your C code in empty box: **`x=`**