2012.1 Final Exam : C Programming

supervisor signature	

Student Id. : ( ) , Name : ( )

1. (1) (10points) Implement **my_strstr** function that works in the same way as the C standard library function **strstr**. You may use **strlen** function in your implementation.

```
/* my_strstr searches for string str2 in string str1. It returns a pointer to the first occurrence of string str2 in string str1, or it returns a null pointer if string str2 is not part of string str1. For example, if str1 is "This is a simple problem", calling my_strstr(str1,"simple") returns a pointer that points to the character 's' in string str1. */
char* my_strstr(char* str1 , char* str2)
{

}
```

(2) (10points) What is the screen output of following C code that uses above **my_strstr** function and standard library functions?

```
int main ()
{
    char str[]="This is a simple problem.";
    char *p;
    p = my_strstr(str,"simple");
    printf("p=%s , str=%s\n",p,str);
    strcpy (p,"complicated");
    str[4]='T';
    printf("p=%s , str=%s\n",p,str);
    return 0;
}
```

Write your answer here

2. (15points) C code below declares matrix variable **mat**, dynamically allocates memory for **mat**, gets keyboard input for N-by-N matrix, computes/prints the sum of all elements in the matrix, and deallocates (free) the memory of the matrix. Fill out the empty boxes (1) and (2) with appropriate C code.

```
#define N 5
int main()
{   float** mat;
    float mat_sum = 0.0;
    int i , j;
    // (1) insert code for dynamic memory allocation for mat (N-by-N matrix)
    for (i = 0 ; i < N ; i++) {
        for (j = 0 ; j < N ; j++) { scanf("%f",&mat[i][j]); mat_sum += mat[i][j]; }
    }
    printf("sum of all matrix elements : %f\n",mat_sum);
    // (2) insert code for memory deallocation (free) for the N-by-N matrix mat
    return 0;
}
```

3. (15points) Suppose **x** is an array of (**int** type) integers, and we have just executed this code:

```
for(i=0;i<5;i++) x[i] = i*i;
```

Suppose that **x[0]** is stored at address **4500**. What is the value of each of the following expressions?

(1) **x** : ( )
(2) ***x** : ( )
(3) **&x[1]** : ( )
(4) **x+2** : ( )
(5) ***(&x[2] +1)** : ( )

4. (10points) What would be printed by the following statements? ( )

```
float *pt;
float a[8]={1.2, 2.3, 3.4, 4.1, 5.2, 2.7, 1.4, 5.9};
pt=&a[1];
pt+=1;
printf("%.1f\n",*pt);
```

5. (15points) What is the screen output of the following code?
( )

```
void f(char *s1, char *s2, int n, int *a)
{
    int i;
    for(i=0; i<n; i++) {
        if(s1[i] > s2[i]) *a = *a+1;
        else if(s1[i] < s2[i]) *(s1+i) = *(s2+i);
        else if(s1[i]== s2[i]) *a = *a-1;
    }
}
```

```
void main()
{
    int a, b;
    char s1[] = "worldcupkorea";
    char s2[] = "koreafighting";
    a = 0;
    f(s1,s2,13,&a);
    printf("%d\n",a);
    printf("s1=%s , s2=%s\n",s1,s2);
}
```

6. (10points) Write a one-line C code that generates a random real number (float type) **x** that is between 6.5 and 7.3 using **rand()** function. Assume **int rand()** function generates a random integer value between 0 and RAND_MAX.

x = ( ) ; **// generate a random real number** $x \in [6.5,7,3]$

7. (15 points) Write the screen output after executing the following program.
( )

```
int func (int a, int b)
{
    a *= 2;
    printf("a = %d, b = %d.\n", a, b);
    return (--a) * (b--);
}

int sub (int *a, int *b)
{
    *b /= 6;
    printf("a = %d, b = %d.\n", *a, *b);
    return (--*a) * (--*b);
}
```

```
int main()
{
    int x = 6, y = 8;
    y = func(x, y);
    printf("x = %d, y = %d.\n", x, y);
    x = sub(&x, &y);
    printf("x = %d, y = %d.\n", x, y);
    return 0;
}
```