ICS'12 Spring Homework 6

I. In the following code, A and B are constants defined with #define:

```
typedef struct{
short x[A][B];          /*Unknown constants A and B*/
    int y;
}str1;

typedef struct{
    char array[B];
    int t;
    short s[B];
    int u;
}str2;

void setVal(str1 *p, str2 *q){
    int v1 = q->t;
    int v2 = q->u;
    p->y = v1 + v2;
}
```

Gcc generates the following code for the body of *setVal*:

```
1    movl 12(%ebp), %eax
2    movl 36(%eax), %edx
3    addl 12(%eax), %edx
4    movl 8(%ebp), %eax
5    movl %edx, 92(%eax)
```

What are the values of A and B? (The solution is unique.)

A: 5
B: 9

II. The following declaration defines a class of structures for use in constructing binary trees:

```
typedef struct ELE *tree_ptr;
struct ELF{
    long val;
    tree_ptr left;
    tree_ptr right;
};
```

For a function with the following prototype:

long trace(tree_ptr tp);

GCC generates the following x86-64 code:

```
1    trace:
       tp in %rdi
2        movl $0, %eax
3        testq %rdi, %rdi
4        je .L3
5      .L5:
6        movq (%rdi), %rax
7        movq 16(%rdi), %rdi
8        testq %rdi, %rdi
9        jne .L5
10     .L3:
11       rep
12       ret
```

A. Generate a C version of the function, using a while loop.

B. Explain in English what this function computes.

A.

```
long trace(tree_ptr tp)
{
    if(NULL == tp)
       return 0;
    while( tp->right )
    {
         tp = tp->right;
    }
    return (tp->val);
}
```

B. Traversing the right child of a binary tree, until it is a leaf node . And return the value of the leaf node.

III. Writing out the following definitions, you can use the variable **a.**

Examples:

An integer : int a;

*A pointer to an integer : int *a;*

1) A pointer to a pointer to an integer: int **a;

2) An array of 10 pointers to integers: int *a[10];

3) A pointer to an array of 10 integers: int (*a)[10];

4) A pointer to a function that takes an integer argument and returns an integer: int (*a)(int);

5) An array of 10 pointers to functions that take an integer argument and return a pointer to an integer: int * (*a[10](int));

IV. The following code transposes the elements of an M × M array, where M is a constant defined by #define:

```
void transpose(Marray_t A) {
  int i, j;
  for(i=0;i<M; i++)
     for(j=0;j<i; j++) {
        int t = A[i][j];
        A[i][j] = A[j][i];
        A[j][i] = t;
     }
}
```

When compiled with optimization level -O2, gcc generates the following code for the inner loop of the function:

```
1   .L3:
2      movl (%ebx), %eax
3      movl (%esi,%ecx,4), %edx
4      movl %eax, (%esi,%ecx,4)
5      addl $1, %ecx
6      movl %edx, (%ebx)
7      addl $52, %ebx
8      cmpl %edi, %ecx
9      jl .L3
```

A. What is the value of M ?

B. What registers hold program values i and j ?

C. Write a C code version of transpose that makes use of the

optimizations that occur in this loop. Use the parameter M in your code rather than numeric constants.

A: M = 13

B: j: %ecx ; i: %edi

C:
```
void transpose(int (*arr)[M])
{
    int i,j;
    int *q = arr[1];     // 取第一行地址
    for(i = 1; i < M; i++)
    {
        int *p = &arr[0][i]; //取第 0 行 i 列地址
        {
            for(j = 0; j < i; j++)
            {
                int temp = q[j];
                q[j] = *p;
                *p = temp;
                p += M;   //转同列下一行
            }
            q += M;        //转下一行首地址
        }
    }
}
```