

Due: 9AM, Monday, November 5 in class

10 points total

Please show your answers (and all work) on these two sheets. Fold your paper the long way and write **your name** and **Homework 7** on the outside.

1. (2 points) Consider the source code below, where M and N are constants declared with **#define**.

```
int mat1[M][N];
int mat2[N][M];

int sum_element(int i, int j)
{
    return mat1[i][j] + mat2[j][i];
}
```

Suppose the above code generates the following assembly code:

```
sum_element:
    pushl   %ebp
    movl   %esp, %ebp
    movl   8(%ebp), %edx
    movl   12(%ebp), %eax
    popl   %ebp
    leal   (%edx,%edx,2), %ecx
    leal   (%eax,%ecx,8), %ecx
    leal   (%eax,%eax,4), %eax
    leal   (%edx,%eax,2), %eax
    movl   mat2(,%eax,4), %eax
    addl   mat1(,%ecx,4), %eax
    ret
```

What are the values of M and N?

M =

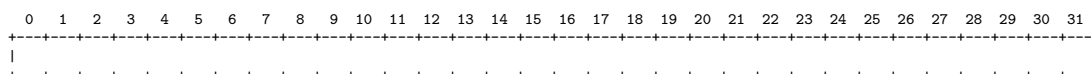
N =

2. (5 points) Consider the following C declaration:

```
struct Node{
    int value;
    char c;
    char d;
    short flag;
    struct Node* left;
    struct Node* right;
    short unit;
    struct Node* next;
};
typedef struct Node* pNode;
/* NodeTree is an array of N pointers to Node structs */
#define N 100
pNode NodeTree[N];
```

2a. Using the template below (allowing a maximum of 32 bytes), indicate the allocation of data for a `Node` struct. Mark off and label the areas for each individual element (there are 8 elements in the struct). Cross hatch the parts that are allocated but not used (to satisfy alignment).

Assume the Linux alignment rules discussed in class. **Clearly indicate the right hand boundary of the data structure with a vertical line.**



- 2b. For each of the three following C references, please indicate which assembly code section (labeled A through F below) places the value of that C reference into register `%ecx`. If no match is found, please write “NONE” next to the C reference.

The register-to-variable mapping for each assembly code section is:

```
%edx = starting address of the NodeTree array
%eax = i (initially)
%ecx = result
```

-----  
C References:

1. ----- NodeTree[i]->left->next->right;
  2. ----- NodeTree[i]->left->left->next;
  3. ----- NodeTree[i]->next->left->right;
- 

Linux/IA32 Assembly fragments:

A:	<pre>movl (%edx,%eax,4),%eax movl 8(%eax),%eax movl 12(%eax),%eax movl 20(%eax),%ecx</pre>	B:	<pre>movl (%edx,%eax,4),%eax movl 8(%eax),%eax movl 20(%eax),%eax movl 12(%eax),%ecx</pre>
C:	<pre>movl (%edx,%eax,4),%eax movl 8(%eax),%eax movl 8(%eax),%eax movl 20(%eax),%ecx</pre>	D:	<pre>movl (%edx,%eax,4),%eax movl 20(%eax),%eax movl 12(%eax),%eax movl 8(%eax),%ecx</pre>
E:	<pre>movl (%edx,%eax,4),%eax movl 12(%eax),%eax movl 8(%eax),%eax movl 20(%eax),%ecx</pre>	F:	<pre>movl (%edx,%eax,4),%eax movl 20(%eax),%eax movl 8(%eax),%eax movl 12(%eax),%ecx</pre>

3. (3 points) In the space provided below, give the output of the C program shown. For information about `strcpy`, see H&S p. 350 or type `man strcpy`. The ASCII values of the characters A through E are 0x41 through 0x45, respectively. Recall that the format string `%08x` specifies that the output item with which it matches should be printed in a field of at least 8 hexadecimal characters wide with leading zeros, if required. The other format strings have similar definitions (see pp. 387-400 of Harbison and Steele for more information).

```
#include <stdio.h>
#include <string.h>

union {
    char c[8];
    short s[3];
    int i[2];
} yew;

int main()
{
    yew.i[0] = 0x12345678;
    yew.i[1] = 0xdeadbeef;
    printf("%04hx %04hx %04hx\n", yew.s[0], yew.s[1], yew.s[2]);
    strcpy(yew.c, "CABABC");
    printf("%08x %08x\n", yew.i[0], yew.i[1]);
    yew.s[0]=0xa1b2;
    yew.s[1]=0xc3d4;
    yew.s[2]=0xe5f6;
    printf("%08x %08x\n", yew.i[0], yew.i[1]);
    return 0;
}
```

OUTPUT: