ECGR 4101/5101 - Assignment 4

1. Let REF(x,i) $--> $ DEF(x.k) denote that the linker will associate an arbitrary reference to symbol x in module i to the definition of x in module k. For each example that follows, use this notation to indicate how the linker would resolve references to the multiply defined symbols in each module. If there is a link time error write "ERROR". If the linker arbitrarily chooses one of the definitions, write "UNKNOWN".

A.

```
/* Module 1 */
int main()
{ }

/* Module 2 */
int main;
int p2()
{ }
```

(a) REF(main.1) —> DEF(___.__)
(b) REF(main.2) —> DEF(___.__)

B.

```
/* Module 1 */
void main()
{ }

/* Module 2 */
int main=1;
int p2();
{ }
```

(a) REF(main.1) —> DEF(___.__)
(b) REF(main.2) —> DEF(___.__)

C.

```
/* Module 1 */
int x;
void main()
```

```
{ }

/* Module 2 */
double x = 1.0;
int p2()
{ }
```

(a) REF(x.1) —> DEF(___.__)
(b) REF(x.2) —> DEF(___.__)

2. Consider the following C codes

```
/* swap.c */
extern int buf[];

int *bufp0 = &buf[0];
static int *bufp1;

static void incr()
{
        static int count = 0;
        count ++;
}

void swap()
{
        int temp;
        incr();
        bufp1 = &buf[1];
        temp = *bufp0;
        *bufp0 = *bufp1;
        *bufp1 = temp;
}

/* main.c */

void swap();

int buf[2] = {1,2};

int main()
{
        swap();
        return 0;
}
```

Compile the two files to generate the object files (gcc -c main.c swap.c). Examine the symbol tables of the object files using readelf utility (readelf –symbols main.o swap.o)

For each symbol that is defined and referenced in swap.o, indicate if it will have a symbol table entry in .symtab section in module swap.o. If so, indicate the module that defines the symbol (swap.o or main.o), the symbol type (local, global, extern), and the section (.text, .data, or .bss) it occupies in that module.
A) buf
B) bufp0
C) bufp1
D) swap
E) temp
F) incr
G) count

Using readelf examine the .data section of main.o (readelf -x .data main.o) to see buf[]. Produce the final object file by linking main.o and swap.o (ld main.o swap.o -o out). Now determine the relocated address of buf[] in the .data section of out. Verify using objdump if the .text section of swap() uses this address when extracting the address of buf in the statement bufp1 = &buf[1].
3. Let the main routine in the .text section of main.o calls the swap routine defined in swap.o. Below is the disassembled listing for the call instruction, as generated by the GNU OBJDUMP tool:

```
6: e8 fc ff ff ff          call 7 <main+0x7> #swap();
                           7: R_386_PC32
```

Suppose that the linker has determined that the ADDR(.text) = 0x80483b4 and ADDR(swap) = 0x80483c8.

Determine the relocated form of the call instruction

```
0x80483ba: e8 -- -- -- --          call ---------- <swap> \\
```