Week 3 Participation

Q1. Practice - Determine the Operation

```
asm
/*
.file"ctest.c"
.text
.globlctest
.typectest, @function
ctest:
.LFB0:
.cfi_startproc     # rsi = b
cmpq%rsi, %rdi # rdi = a
setg%r10b
cmpl%edi, %esi
setl%al
cmpl%edi, %edx
setnb%r9b
cmpq%rdx, %rdi # rdx = c
setne%r8b
cmpb%sil, %dl  # dl = c
seta%cl        # sil = b
testq%rdi, %rdi # a & a
setne%dl
addl%r10d, %eax
addl%r9d, %eax
addl%r8d, %eax
addl%ecx, %eax
addl%edx, %eax
ret
.cfi_endproc
.LFE0:
.sizectest, .-ctest
.ident"GCC: (GNU) 4.8.5 20150623 (Red Hat 4.8.5-4)"
.section.note.GNU-stack,"",@progbits

*/
```

asm

```
c
/* fill in the blanks based on the asm code */

char ctest(long a, long b, long c) {
  char t1 = a __?__ b;
  char t2 = (int)a __?__ (int) b;
  char t3 = (unsigned)c __?__ (unsigned) a;
  char t4 = a __?__ c;
  char t5 = (unsigned char)c __?__ (unsigned char)b;
  char t6 = a __?__ 0;
  return t1+t2+t3+t4+t5+t6;
}
```

Operator used to calculate `t1`:Operator used to calculate `t2:Operator used to calculate `t3`:Operator used to calculate `t4`:Operator used to calculate `t5`:Operator used to calculate `t6`:

Q2. Practice - Loops using `goto`s

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

```
int sum(int *a, int len)
{
  int sum = 0;
  for (i = 0; i < len; i++)
  {
      sum+= a[i];
  }
  return sum;

}

int a[] = {1, 2, 5, 7, 4, 6, 8, 0, 11};

int main()
{
  int v = sum(a, sizeof(a) / sizeof(int));

  printf("%d\n", v);

  return 0;
}
```

Re-implement `sum()` from the above program.
- using `goto` and labels
- without using `for`, `while`, or `do`
- with only `if`s without `else`s

Q3. Practice - Interpreting Assembly

```
/* Given the assembly code representation, Determine
 * which variables each of the registers is used for.
 */

int search(int *a, int size, int find_v) {
  int lo = 0, hi = size;

  while (lo < hi)  {
    int mid = (lo + hi) / 2;
    if (a[mid] < find_v)
      hi = mid;
    else if (a[mid] > find_v)
      lo = mid+1;
    else
      lo = hi = mid;
  }

  return lo;
}


.file   "bin.c"
    .text
    .globl  search
    .type   search, @function
search:
.LFB0:
    .cfi_startproc
    movl    $0, %eax
    jmp     .L2
.L4:
    leal    (%rax,%rsi), %ecx
    movl    %ecx, %r8d
    shrl    $31, %r8d
    addl    %r8d, %ecx
    sarl    %ecx
    movl    %ecx, %r9d
    movslq  %ecx, %r8
    movl    (%rdi,%r8,4), %r8d
    cmpl    %edx, %r8d
    jl      .L3
    jle     .L5
    leal    1(%rcx), %eax
    movl    %esi, %r9d
    jmp     .L3
.L5:
    movl    %ecx, %eax
.L3:
    movl    %r9d, %esi
.L2:
    cmpl    %esi, %eax
    jl      .L4
    rep ret
    .cfi_endproc
.LFE0:
    .size   search, .-search
    .ident  "GCC: (GNU) 4.8.5 20150623 (Red Hat 4.8.5-4)"
    .section    .note.GNU-stack,"",@progbits
```

`%eax` is used for variable:`%rdi` is used for variable:`%esi` is used for variable:`%edx` is used for variable:
`%ecx` is used for variable:`%r9d` is used for variable:`%r8` is used for variable: