CS 304
Homework 4

How to submit --

Save all your answers to hw4.txt and type the following to submit after you finish all the problems:

```
~lyang11/bin/submit cs304 hw4 hw4.txt
```

1. Consider the following C functions and assembly code:

```c
int fun1(int a, int b)
{
   return (a >= b) ? b : a;
}

int fun2(int a, int b)
{
   unsigned int c = b;
   if (a >= c)
      return c;
   else
      return a;
}

int fun3(int a, int b)
{
   if (b < a)
      return b;
   else
      return a;
}
```

```
      pushq   %rbp
      movq    %rsp, %rbp
      movl    %edi, -4(%rbp)
      movl    %esi, -8(%rbp)
      movl    -8(%rbp), %eax
      cmpl    -4(%rbp), %eax
      jge     .L2
      movl    -8(%rbp), %eax
      jmp     .L3
.L2:
      movl    -4(%rbp), %eax
.L3:
      popq    %rbp
      ret
```

Which of the functions compiled into the assembly code shown?

2. Consider the following C functions and assembly code:

```
int fun4(int a)
{
   return a * 14;
}


int fun5(int a)
{
   return a * 56;
}


int fun6(int a)
{
   return a * 28;
}
```

```
        pushq   %rbp
        movq    %rsp, %rbp
        movl    %edi, -4(%rbp)
        movl    -4(%rbp), %eax
        sall    $2, %eax
        leal    0(,%rax,8), %edx
        movl    %edx, %ecx
        subl    %eax, %ecx
        movl    %ecx, %eax
        popq    %rbp
        ret
```

Which of the functions compiled into the assembly code shown?

3. Consider the following assembly code for a C for loop:

```
        cmpl    %esi, %edi
        jle     .L4
        movl    $1, %eax
.L3:
        addl    %edi, %eax
        subl    $1, %edi
        cmpl    %esi, %edi
        jne     .L3
        ret
.L4:
        movl    $1, %eax
        ret
```

Based on the assembly code above, fill in the blanks below in its corresponding C source code. (Note: you may only use the symbolic variables i, x, y, and r in your expressions below --- do not use register names.)

```
int loop(int x, int y)
{
  int i;
  int r = ________________________;

   for (i=_________; ____________________________; ___________) {

       ________________________________________________;

   }

   return r;

}
```

4. Each of the assembler routines on the left were created by applying gcc -S -O to a C source program. Match each assembler routines on the left with the equivalent C function.

```
func7:
        movl    $0, %eax
        ret

func8:
        movl    %edi, %eax
        sarl    $31, %eax
        ret

func9:
        movl    %edi, %eax
        andl    $-2147483648, %eax
        ret
```

```
int choice1(int x)
{
  return (x>>31)<<31;
}
```

```
int choice2(int x)
{
  unsigned u = x;
  return u>>31;
}

int choice3 (int x)
{
  return (x<<31)>>31;
}

int choice4(int x)
{
  return (x < 0U);
/*Here 0U is an unsigned integer 0. x<0U automatically change x to unsigned number for this line fo code
*/
}

int choice5(int x)
{
  return x>>31;
}

int choice6(int x)
{
  return (x < 0);
}
```

Fill in your answers here: (enter ``none'' if no choices match)

fun7 corresponds to choice

fun8 corresponds to choice

fun9 corresponds to choice

5. Convert this function into pointer-based code.

```
    void shift(int a[], int n) {
      int i;
      for(i = 0; i != n-1; i++)
        a[i] = a[i+1];
    }
```

6. Pointers

On the left is a short C program (blocks.c) that uses a series of operations involving pointers. Fill in the blanks on the right with the value of the requested variable AFTER the execution of the instruction across from it (use char notation for characters and hex for addresses). Assume the address of the blocks array is 0x4680.

Note: Make sure you do this by hand at first. The point here is to learn how pointers and pointer statements work.

```
int main(void) {
   char blocks[3] = {'A','E','P'};
   char *ptr = &blocks[0];
   char temp;

   temp = blocks[0];
   temp = *(blocks + 2);
   temp = *(ptr + 1);
   temp = *ptr;

   ptr = blocks + 2;
   temp = *ptr;
   temp = *(ptr - 1);

   ptr = blocks;
   temp = *++ptr;
   temp = ++*ptr;
   temp = *ptr++;
   temp = *ptr;

   return 0;
}
```

```
  blocks = 0x4680
  ptr = _____
  temp = _____

  temp = _____
  temp = _____
  temp = _____
  temp = _____

  ptr = _____
  temp = _____
  temp = _____

  ptr = _____
  ptr = _____, temp = _____
  ptr = _____, temp = _____
  ptr = _____, temp = _____
  temp = _____
```