HW7

How to submit:

Save all your answers to hw6.txt and type the following to submit after you finish all the problems:

~lyang11/bin/submit cs304 hw7 hw7.txt

1. Simple program

I wrote the following programs named test.c, then compiled and ran using
gcc -o test test.c
./test

First program:

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

int j=2;

main()
{
  char a="abcdefg";
  int i=1;
  int *p;
  p=&i;
  p[1]=7;
  a[7]='3';
  a[8]='3';
  a[9]='3';
  i++;
  j++;
  printf("%s %d %d\n",a, i, j);
  return 0;
}

---------------------------------
```

When I compile it, it gives errors. What happened?

Then I changed the program:

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

int j=2;

main()
{
  char *a="abcdefg";
  int i=1;
  int *p;
```

```
  p=&i;
  p[1]=7;
  a[7]='3';
  a[8]='3';
  a[9]='3';
  i++;
  j++;
  printf("%s %d %d\n",a, i, j);


  return 0;
}
```

When I run it, it gives segmentation fault. Use GDB to find out what happened.

Another try:

```
#include <stdio.h>

int j=2;

main()
{
  char a[]="abcdefg";
  int i=1;
  int *p;
  p=&i;
  p[1]=7;
  a[7]='3';
  a[8]='3';
  a[9]='3';
  i++;
  j++;
  printf("%s %d %d\n",a, i, j);


  return 0;
}
```

I run it again. It does not print out a at all. What happened? Use GDB to figure out what happened.

2. Implicit list header

Determine the block sizes and header values that would result from the following sequence of malloc requests. Assumptions: (1) The allocator maintains double-word alignment, and uses an implicit free list with block format from Fig. 9.35 in the book (pp.821). (2) Block sizes are rounded up to the nearest multiple of 8 bytes.

Note that the format of the header is 32 bits: $b_{31}b_{30}b_{29}...b_4b_300a$
where $b_{31}b_{30}b_{29}...b_4b_3000$ will be the block size and the last bit "a" shows whether the block has been allocated (a=1 allocated, a=0, free)

Request Block size (decimal bytes) Block header (hex)

malloc(3) _____ _____
malloc(11) _____ _____
malloc(20) _____ _____
malloc(21) _____ _____

3. Next-fit

Implement a find_fit function for the simple alocator described here.

static void *find_fit(size_t asize)

Yur solution should perform a **next-fit search** of the implicit free list. Next-fit should search till the end of the heap and continue to the beginning of the heap.

All the related programs in the book can be found here (including memlib.c and mm.c -- before you work on the next project (lab5 - malloclab), you should understand each line of the code well). Notice that the function mm_malloc() calls the function find_fit().