This exam is closed-book, closed-notes. Answer all questions in the exam booklet.

Preliminaries (1 minute, 1 point)

A. Print your name on your exam booklet.
B. Print your netid next to your name.
C. `printf("Precept %s", your_precept_number)` on your exam booklet (refer to the table below).
D. Write and sign the honor pledge.

Precept sections:

1  MW 1:30  Gabai  
2  MW 3:30  Gabai  
3  MW 7:30  Kyung  
4  TTh 12:30  Li  
4A TTh 12:30  Mizrahi  
5  TTh 1:30  Li  
5A  TTh 1:30  Zhang  
6  TTh 3:30  Le  
6A TTh 3:30  Stafman  

In answering the questions, assume the C compiler is the same `gcc217` you have been using on `courselab`.

Just in case you’ve forgotten:

```c
int isspace(int c);
/* returns 1 if c is a whitespace character;
   returns 0 if c is EOF or any nonwhitespace ASCII character. */

int isdigit(int c);
/* returns nonzero if c is a decimal digit;
   returns 0 if c is EOF or any other ASCII character. */
```
1. Write a function-header comment for this `read_digits` function, describing what it does.

```c
int next; /* This variable holds the next character from stdin, 
           after it has been read but before it is fully processed. */

int read_digits (char a[], int n) {
    int i=0; int status;
    if (!isdigit(next))
        return 0;
    while (isdigit(next)) {
        if (i<n)
            a[i++]=next;
        next = getchar();
    }    
    if (i<n)
    {a[i]='$'; status = 1;}
else status = -1;
return status;
}
```

The rest of the questions concern the program shown on the next page.

2. Write a function-header comment for function `read_while`.  
   (Hint: it’s just like `read_digits`, except for the function-parameter `f`, and the assertions.)

3. Write a function-header comment for function `main`.

4. True or false: Some input string (on stdin) can cause the assertion at line 34 to fail.  
   If true, show the input.

5. True or False: There is an input to this program that will cause signed integer overflow at line 21.  
   If true, show the input.

6. True or False: There is an input to this program that will cause signed integer overflow at line 36.  
   If true, show the input.

7. True or False: There is an input to this program that will cause signed integer overflow at line 47.  
   If true, show the input.

8. Write the function `read_while_entry` that contains function-entry assertions for `read_while`.  
   It should be as strong (detect as many erroneous conditions) as you can reasonably make it.

   ```c
   void read_while_entry(int (*f)(int), char a[], int n) { ... }
   ```

9. Write a `main` function, as simple as possible, that calls `read_while` with an erroneous condition that is not caught by your `read_while_entry` assertions. (Your `main` function doesn’t have to do anything useful; it doesn’t have to match in any way the behavior of my main function.)

10. Write the function `read_while_exit` that contains function-exit assertions for `read_while`.  
    It should be as strong as you can reasonably make it.

    ```c
    void read_while_exit(int (*f)(int), char a[], int n, int i, int status) { ... }
    ```

    Hint: think about all the relationships that should hold between n, i, status, and the contents of a.
```c
#include <stdlib.h>
#include <stdio.h>
#include <ctype.h>
#include <string.h>
#include <assert.h>

int next; /* This variable holds the next character from stdin,
after it has been read but before it is fully processed. */

void read_while_entry(int (*f)(int), char a[], int n) { }
void read_while_exit(int (*f)(int), char a[], int n, int i, int status) { }

int read_while (int (*f)(int), char a[], int n) {
  int i=0; int status;
  /* Hint: f might be isspace, isdigit, isalpha, etc. */
  read_while_entry(f,a,n); /* assertions for function entry */
  if (!f(next))
    return 0;
  while (f(next)) {
    if (i<n)
      a[i++]=next;
    next = getchar();
  }
  if (i<n)
    {a[i] = '\0'; status = 1;}
  else status = -1;
  read_while_exit(f,a,n,i,status); /* assertions for function exit */
  return status;
}

/* Convert the null-terminated string [a] from ASCII decimal to int. */
int convert (char *a) {
  char *p; int s=0;
  assert (strlen(a) <= 9);
  for (p=a; *p; p++)
    s = s*10 + (*p-'0');
  return s;
}

int main (void) {
  int i, m=0, stat; char a[10];
  next=getchar();
  for(i=0; i<4; i++) {
    read_while(isspace, NULL, 0);
    stat=read_while(isdigit, a, 10);
    if (stat==1)
      m += convert(a);
    else exit(1);
  }
  printf("%d\n",m);
  return 0;
} 
```