• This examination is closed-book and closed-notes. You have 50 minutes to complete it. The total number of points is 50 as well.
• Please answer all questions. The exam is printed two-sided, so please make sure you don’t miss any questions.
• Please read through all questions before starting to answer any. It is advisable that you determine which questions you are most comfortable with or that will be easiest for you to answer correctly, and do those first.
• Please make sure you read and sign the honor code below.
• PLEASE WRITE YOUR NAME AND YOUR PRECEPTOR’S NAME CLEARLY.

Name:

Preceptor’s Name:

I pledge my honor that I have not violated the honor code during this examination.

___________________

For instructor use only:

<table>
<thead>
<tr>
<th>Question</th>
<th>Max Points</th>
<th>Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>II</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>III</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>IV</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>V</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>50</td>
<td></td>
</tr>
</tbody>
</table>
Question I: (10 points)

(a) (1 point) True or False: A break statement transfers control over the nearest surrounding while, for, switch or if construct. Explain.

(b) (1 point) What is the difference between the following two declarations:

```c
typedef struct listnode *Node1, *Node2;
typedef struct listnode* Node1, Node2;
```

(c) (1 points) Is the following code good or not good. Say why.

```c
main() {
    int *p;
    *p = 1;
}
```

(d) (2 points) Consider the following declaration:

```c
int a[10];
```

(i) What is the type of the expression (a+1)?

(ii) Express a[1] in terms of a+1.
(e) (2 points) With regard to good practices in defining makefiles that we discussed in lecture, what are the problems with the following dependence line: (Note that it is just a dependence line; do not worry about the command line.)

```
  test_all.o:  test_all.c test_one.c my_library.o
```

(f) (1 point) True or False: It is a benefit of using make that if test_all.c includes test_all.h using `#include`, then test_all.o has to express a dependence only on test_all.c and not on both test_all.c and test_all.h.

(g) (2 points) You have written a module to implement a queue, and published an interface for it in a .h file. A colleague reviews it and suggests you add a function to the interface. What are the two questions you ask yourself to decide whether or not to include the new function in the module interface?
Question II (10 points):

(a) Consider the code

```c
float *p;
p = p + 4;
```

(i) (1 point) If p had the value 3680 before the second statement above, what would its value be after the second statement, using gcc217 on our nobel systems?

(ii) (1 point) If you wanted to ensure that the value after that statement is 3684, what statement would you execute instead of p = p + 4, using p as the only variable and 4 as the only constant in your expression?

(b) (1 point) The following function takes an array of integers as a parameter:

```c
int f(int a[]);
```

Show another way to declare the function such that an array of integers can be passed to it.

(c) (1 point) What **warnings** or **errors** does the command “gcc217 –c –O2 test.c ” generate when invoked on the file test.c below. If it generates a warning rather than an error, say why?

```c
#include <string.h>

int main(void) {
    char *a = “hello”;
    printf(“%s
”, a);
}
```
(d) (4 points) Consider the following code:

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

int * printInt(const int *);

int main() {
  int a[10], i, *p, *a1;
  p = &i;
  a[5] = 20;
  p[0] = 1;
  p[5] = 20;
  a = a+1;
  a1 = malloc(10*sizeof(int));
  realloc(a1,100*sizeof(int));
  a = a1;
  p = a;
  p++;
}
```

For each of the above statements numbered 1-11, put a letter next to it as follows. In each case where the letter is (B) - (D), say why.

- (A) if it is okay
- (B) if it is okay but is either not meaningful as written or should be better expressed
- (C) if it will generate a compile-time error
- (D) if it could cause a run-time error (either definitely or by doing something dangerous)

(e) (2 points) State the difference between the two directives:

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

int * printInt(const int *);

int main() {
  int a[10], i, *p, *a1;
  p = &i;
  a[5] = 20;
  p[0] = 1;
  p[5] = 20;
  a = a+1;
  a1 = malloc(10*sizeof(int));
  realloc(a1,100*sizeof(int));
  a = a1;
  p = a;
  p++;
}
```

Give one example of a line that could precedes these, such that one of the preprocessor tests above is satisfied and the other one is not.
Question III: (10 points)

(a) In lecture, we discussed the use of generic functions in C. We described how to pass a comparison function to a Stack module, and we discussed that the client could define the comparison function as follows:

```c
int strCompare(const void *item1, const void *item2) {
    char *str1 = item1;
    char *str2 = item2;
    return strcmp(str1, str2);
}
```

and pass it to the `Stack_areEqual()` function that is part of the Stack interface as follows:

```c
if (Stack_areEqual(s1,s2,strCompare)) { …}
```

(i) (2 points) If we do it this way, what does corresponding parameter of `Stack_areEqual()` look like? That is, fill in the third parameter in the declaration of `Stack_areEqual` below:

```c
int Stack_areEqual(Stack_T s1, Stack_T s2,
```

(ii) (1 point) Why do we have to define `StrCompare`, rather than just pass the library function `strcmp()` as follows:

```c
if (Stack_areEqual(s1,s2,strcmp)) { …}
```

(iii) (2 points) Could the client pass `strcmp` as the third parameter? Show the resulting call to `Stack_areEqual()`?
(b) Consider the following program.

```c
int main(void) {
    int a = 5; int b = 1;
    /* insert function call to IncrementAndSwap() here */
    printf("a = %d\nb = %d\n", a,b);
    return 0;
}
```

(i) (3 points) Write a function called IncrementAndSwap(...), which takes two parameters, such that the above program prints:

```
a = 2
b = 6
```

That is, IncrementAndSwap() should effectively first increment the values of a and b and then Swap them. IncrementAndSwap() should call a function called Swap(...), which is defined as follows:

```c
static void Swap(int *x, int *y) {
    int temp;
    temp = *y;
    *y = *x;
    *x = temp;
}
```

Also, write the call to IncrementAndSwap() at the place shown marked with a comment in the main() program above.
(ii) (2 points) Suppose the formal parameters of `IncrementAndSwap()` are called \( x \) and \( y \) as well. What are the values of \( x \) and \( y \) at the point just before the first statement within `IncrementAndSwap()` is executed? Let’s call this “at Point 1.” What are the values of \( x \) and \( y \) when the `printf()` function in `main()` above is invoked? Let’s call this “at Point 2.”

At Point 1:

At Point 2:
Question IV: (10 points)

(a) (1 point) What does the use of “const” in the declaration of the function \( f() \) below say about what \( f() \) can and cannot do?

\[
\text{int } f(\text{const struct node } *\text{node}_1);
\]

(b) (2 points) What is the risk with having the following line of code in a C program:

\[
\text{assert (} f() == 0);\]

How would you rewrite the code to eliminate the risk?

(c) (1 point) How should we define functions that are not in the interface of a module (e.g. SymTable\_hash()), and why?
(d) (6 points) Draw a DFA that accepts legal strings defined as follows (and only such legal strings):
A legal string is a string that begins with 1 to 3 a's, is followed by 0 or more b's or c's, and ends with at least one d.
Question V: (10 points)

(a) (2 points) You are asked to implement several versions of the function `strcpy(char *dest, char *source)`. The first version requires you to use an infinite loop. Fill in the blank lines marked `XX` with code that will do this. Each line marked `XX` should need one line of (well-styled) C code.

```c
void strcpy1(char dest[], const char source[]) {
    int i = 0;
    XX
        dest[i] = source[i];
    XX
        i++;
    }
}
```

(b) (2 points) The next version asks you to move the assignment from `source` to `dest` into a condition test. Fill in the code to do this below:

```c
void strcpy2(char dest[], const char source[]) {
    int i = 0;
    XX
        i++;
    }
}
```

(c) (3 points) The third version asks you to get rid of the variable `i`, and implement the function in a single statement (with no nested statements). Write that line of code below.

(d) You profile your program and find that half its time is spent in one loop. You optimize that loop and get it to take one-fourth the time it used to take, for the same input data set.

(i) (2 points) By what fraction did you reduce the execution time of the program?
(ii) (1 point) Your friend ran the old program and your new, optimized program and said that the new program produced only a five percent reduction in execution time. What do you think would be the main reason for this?