This midterm test consists of 4 questions on 8 pages (including this one). When you receive the signal to start, please make sure that your copy of the test is complete. Extra space was left for each of the programming questions. Please indicate clearly the part of your work that should be marked.

IMPORTANT: Assume all #includes have been done for you. You do not need to perform error checking unless the question specifically asks you to.

Marking Guide

# 1: _____/ 6
# 2: _____/ 6
# 3: _____/ 9
# 4: _____/ 8
TOTAL: _____/29

Good Luck!
Question 1.  [6 marks]
Given the following struct definition:

```c
struct node {
    int v;
    struct node *next;
};
```

Part (a)  [2 marks]
Explain and fix the error in the following code:

```c
void f(struct node *n, int v) {
    n->v = v;
    n->next = NULL;
}
int main() {
    int value = 10;
    struct node *newnode;
    f(newnode, value);
    return 0;
}
```

No memory is allocated for newnode. It can be fixed by adding "newnode = malloc(sizeof(struct node));" after the declaration of newnode in main.

Part (b)  [2 marks]
Explain and fix the error in the following code:

```c
void g(struct node *n, int v) {
    n = malloc(sizeof(struct node));
    n->v = v;
    n->next = NULL;
}
int main() {
    int value = 10;
    struct node *newnode;
    g(newnode, value);
    return 0;
}
```

When g returns, the value of newnode will not have changed. In other words changing the value of n inside g does not propagate out of the function. There are several ways to fix this. One is to return n from g.

Part (c)  [2 marks]
Explain and fix the error in the following code: Bad copy and paste error. The code should have been:

```c
struct node *h(int v) {
    struct node n;
    n.v = v;
    n.next = NULL;
    return &n;
}
```
void g(struct node *n, int v) {
    n = malloc(sizeof(struct node));
    n->v = v;
    n->next = NULL;
}

int main() {
    int value = 10;
    struct node *newnode;
    newnode = h(value);
    return 0;
}

h returns a pointer to memory that is allocated on the stack. When h returns this memory is no longer reserved for n, so the values at that address will change. A solution is to use malloc inside h to allocate memory for the node on the heap.
Question 2.  [6 marks]
Part (a)  [5 marks]
Fill in the type for the following variable declarations so that the variables have the correct type.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>int i;</td>
</tr>
<tr>
<td>2</td>
<td>________ q = &amp;i;</td>
</tr>
<tr>
<td>3</td>
<td>________ j = *q;</td>
</tr>
<tr>
<td>4</td>
<td>char a[10];</td>
</tr>
<tr>
<td>5</td>
<td>________ c = a;</td>
</tr>
<tr>
<td>6</td>
<td>________ b = &amp;a[1];</td>
</tr>
<tr>
<td>7</td>
<td>________ d = a[3];</td>
</tr>
<tr>
<td>8</td>
<td>char *strs[5];</td>
</tr>
<tr>
<td>9</td>
<td>________ s = strs;</td>
</tr>
<tr>
<td>10</td>
<td>________ t = &amp;strs[3];</td>
</tr>
<tr>
<td>11</td>
<td>________ u = strs[3];</td>
</tr>
<tr>
<td>12</td>
<td>________ v = strs[3][1];</td>
</tr>
<tr>
<td>13</td>
<td>________ w = *strs[3];</td>
</tr>
</tbody>
</table>

Part (b)  [1 mark]
Write down the line numbers that could lead to a memory error (segmentation fault or bus error).
12, 13

Solutions:
```
int i;
int *q = &i;
int j = *q;
char a[10];
char *cptr;
char *c = a;
char *b = &a[1];
char d = a[3];
char e = a[3];
char *strs[5];
char **s = strs;
char **t = &strs[3];
char *u = strs[3];
char v = strs[3][1];
char w = *strs[3];
```
Question 3.  [9 marks]

Complete the C function below that removes characters in the string ch from the string str. You may not use any string library functions other than strlen.

Below is an example of calling strip:

```c
char input[20] = "Hello there";
strip(input, "eo");
printf("%s", input); // prints "Hll thr"
```

```c
void strip(char *str, char *ch) {
    int i = 0;
    int k = 0;
    while(i < strlen(str)) {
        int j = 0;
        while(j < strlen(ch) && ch[j] != str[i]) {
            j++;
        }
        if(j >= strlen(ch)) {
            str[k] = str[i];
            k++;
        }
        i++;
    }
    str[k] = '\0';
}
```

There are several different ways to solve this problem.
Question 4. [8 marks]

Part (a) [1 mark]
Rewrite the following shell loop so that no additional processes are created:

```bash
for f in `ls /tmp`
  for f in /tmp/*
```

Part (b) [4 marks]
Complete the shell function below that replaces spaces in a string with underscore characters and prints the resulting string. Do not use `tr`, `sed`, or `awk`. Use shell commands we discussed in class such as `set` or `cut`. You will get 3 marks for successfully replacing only the first space. If a string does not have a space the function will print the original string.

For example, calling

```
replace_space "a string with spaces"
```

will produce

```
a_string_with_spaces
```

```bash
function replace_space {
  function rmsp {
    set $1
    result=""
    for str in $*
    do
      if [ -z $result ]
      then
        result=$str
      else
        result=$result"_"$str
      fi
    done
    echo $result
  }

  for f in $1/*
  do
    rmsp "$f"
  done
}```
C function prototypes and structs:

- `int sprintf(char *s, const char *format, ...)`
- `char *strchr(const char *s, int c)`
- `size_t strlen(const char *s)`
- `char *strncat(char *dest, const char *src, size_t n)`
- `int strncmp(const char *s1, const char *s2, size_t n)`
- `char *strncpy(char *dest, const char *src, size_t n)`
- `char *strrchr(const char *s, int c)`
- `char *strstr(const char *haystack, const char *needle)`

Shell comparison operators:

<table>
<thead>
<tr>
<th>Shell</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-d filename</code></td>
<td>Exists as a directory</td>
</tr>
<tr>
<td><code>-f filename</code></td>
<td>Exists as a regular file.</td>
</tr>
<tr>
<td><code>-r filename</code></td>
<td>Exists as a readable file</td>
</tr>
<tr>
<td><code>-w filename</code></td>
<td>Exists as a writable file.</td>
</tr>
<tr>
<td><code>-x filename</code></td>
<td>Exists as an executable file.</td>
</tr>
<tr>
<td><code>-z string</code></td>
<td>True if empty string</td>
</tr>
<tr>
<td><code>str1 = str2</code></td>
<td>True if str1 equals str2</td>
</tr>
<tr>
<td><code>str1 != str2</code></td>
<td>True if str1 not equal to str2</td>
</tr>
<tr>
<td><code>int1 -eq int2</code></td>
<td>True if int1 equals int2</td>
</tr>
<tr>
<td><code>-ne, -gt, -lt, -le</code></td>
<td>For numbers</td>
</tr>
<tr>
<td><code>!=, &gt;, &gt;=, &lt;, &lt;=</code></td>
<td>For strings</td>
</tr>
<tr>
<td><code>-a, -o</code></td>
<td>And, or.</td>
</tr>
</tbody>
</table>

Useful shell commands:

du, echo, ls, head, tail, read, sort, uniq, wc

grep (returns 0 if match is found, 1 if no match was found, and 2 if there was an error)

set (Note: with no arguments set prints the list of environment variables) ps aux - prints
the list of currently running processes grep -v displays lines that do not match