Answer all questions in the space provided. Write clearly and legibly, you will 
not get credit for illegible or incomprehensible answers. This is a closed book 
exam. However, each student is allowed to bring one page of notes to the exam. 
Print your name at the top of every page.

<table>
<thead>
<tr>
<th>Question</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Points</td>
<td>16</td>
<td>16</td>
<td>15</td>
<td>6</td>
<td>7</td>
<td>10</td>
<td>6</td>
<td>9</td>
<td>15</td>
<td>100</td>
</tr>
<tr>
<td>Score:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1. Short answer

(a) What expression should be placed inside the call to malloc to allocate memory for an 
double array with n rows of m columns?

double* x = malloc( ? );

(b) The C preprocessor lets you define macros. They can be very handy but also possible 
cause problems. Why is the following not a good macro?

```
#define SQUARE(x) x*x
```

(c) What is the one pointer type that we can not directly perform arithmetic on, and why is 
it not possible?

(d) It is absolutely essential to free up memory after you are using dynamically allocated 
structures. If you don’t you’ll have memory leaks. Obviously the free function call is used 
for this purpose. So, if you see the following code in a program:

```
free(treenode);
```

and the program segfaults on that line – what might be wrong? Name at least two things.
2. More short answer
   (a) There is no “real” boolean datatype in C, so what is used instead? Please give an example. (4)

   (b) Assuming x has been previously declared and initialized — Is the following code snippet valid C source code? If not, why not? If so, what is the result when this code is executed?

   ```c
   if ( x = 0 )
     printf ( "x is zero\n" );
   ``` (4)

   (c) Is the following snippet valid C source code? If not, why not? If so, what is the result when this code is executed?

   ```c
   int a[] = {3, 4, 5};
   printf("%d\n", 2[a]);
   ``` (4)

   (d) As part of the Huffman coding assignment, you had to implement a priority queue. What structure did you use to implement it? What is another possible way? Which of the two methods would have better performance as I increase the number of possible items placed in the priority queue? (4)
3. What is the output of this program?

Hint: the \%x format specifier prints an integer in hexadecimal format (with lowercase letters), the # says to prefix the result with 0x, and the 04 says to make the output at least 4 characters wide, padding with leading zeros if needed. So, for example, the statement printf("%#04x", 13); would result in 0x0d being printed to standard out.

```c
#include <stdio.h>

void main(void)
{
    unsigned char x = 42;
    unsigned char y = 14;
    unsigned char z = 98;

    unsigned char a = x << 3;
    unsigned char b = x >> 3;
    unsigned char c = x & y;
    unsigned char d = x & z;
    unsigned char e = x | y;
    unsigned char f = x ^ y;

    printf("x %#04x, %d\n", x, x);
    printf("y %#04x, %d\n", y, y);
    printf("z %#04x, %d\n", z, z);

    printf("a %#04x, %d\n", a, a);
    printf("b %#04x, %d\n", b, b);
    printf("c %#04x, %d\n", c, c);
    printf("d %#04x, %d\n", d, d);
    printf("e %#04x, %d\n", e, e);
    printf("f %#04x, %d\n", f, f);
}
```
4. Consider the following code.

```c
void main ( void )
{
    unsigned long a[] = {12, 23, 34, 45, 56};
    unsigned long *x = a;
    printf("sizeof(unsigned long)=%lu ", sizeof(unsigned long));
    printf("x=%p, x[0]=%lu\n", x, x[0]);
    x = x + 3;
    printf("x=%p, x[0]=%lu\n", x, x[0]);
}
```

If the output from lines 5 and 6 is

```
sizeof(unsigned long)=8  x=0x7fff78e71c70, x[0]=12
```

what is the output from line 8?

5. What is the output of this program?

```c
#include <stdio.h>

void main(void)
{
    char data[] = "SPRING";
    data[5] = 'T';
    char *linePt = &data[2];
    *linePt = 'L';
    printf("[%s], [%s]\n", data, linePt);
}
```
The following program is compiled and run with the command: 

```
./a.out 1203201
```

What is the output?

```c
#include <stdio.h>

int main(int argc, char* argv[])
{
    char* c_pt;
    int n = 0;
    if(argc == 2)
    {
        c_pt = argv[1];
        while(*c_pt)
        {
            printf("%c, %d\n", *c_pt, n);

            if(*c_pt < '0' || *c_pt > '2') break;
            n = n*3 + *c_pt - '0';
            c_pt++;
        }
        printf("value = %d\n", n);
    }
}
```
7. What is the output of this program?

```c
#include <stdio.h>

struct Point {
    int x;
    int y;
};

struct Point incPoints(struct Point *p1, struct Point p2) {
    (*p1).x ++;
    p1->y ++;
    p2.x ++;
    p2.y ++;
    return p2;
}

void main(void) {
    struct Point a = {1, 2};
    struct Point b = {3, 4};
    struct Point c = incPoints(&a, b);

    printf("a=(%d, %d)\n", a.x, a.y);
    printf("b=(%d, %d)\n", b.x, b.y);
    printf("c=(%d, %d)\n", c.x, c.y);
}
```
8. For the given tree, write out the following traversals.

(a) Breadth First (also known as level-order):

(b) Depth First, pre-order:

(c) Depth First, post-order:
9. Remember `uuencode`? As part of this binary to text encoding, every three bytes of the binary data is split into four 6 bit groupings. (There’s more to the encoding than that, but I’m not going to have you deal with the rest of it on this exam.)

<table>
<thead>
<tr>
<th>Original bytes, decimal</th>
<th>67</th>
<th>97</th>
<th>116</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original bytes, hex</td>
<td>43</td>
<td>61</td>
<td>74</td>
</tr>
<tr>
<td>Binary</td>
<td>0 1 0 0 0 0</td>
<td>1 1</td>
<td>0 1 1 0 0 0 1</td>
</tr>
<tr>
<td>New hex values</td>
<td>10</td>
<td>36</td>
<td>05</td>
</tr>
<tr>
<td>New decimal values</td>
<td>16</td>
<td>54</td>
<td>5</td>
</tr>
</tbody>
</table>

Write a function that takes an array of three bytes and returns a newly allocated array containing the regrouped bits.