Consider the following variable declarations:

```c
int a;     char b;     double c;
int *p;    char *q;    double **r;
```

State the **type** of each of the following expressions. (Assume all pointers point to something valid.) If any of these expressions would cause a compile error, explain why in a few words. (10 points)

- `a`
- `&b`
- `*q`
- `p[3]`
- `*&c`
- `**p`
- `p + 5`
- `**r`
- `--q`
- `*c`
For each of the code fragments below, write what the program displays in the box to the right.
(10 points)

```c
int r=10, s=20, t=30;
int *x=&r, *y=&t;
int **p=&x;
printf("%d %d %d\n",*y,*x,**p);
x = &s;
*y = 40;
**p = 50;
printf("%d %d %d\n",r,s,t);
```

```c
char word[] = "kookaburra";
char *p = &word[5];
printf("%s\n",p);
for(i=0;i<5;i++) {
    p[i] = word[i];
}
printf("%s\n",word);
```

```c
int mystery( int a, int b, int c )
{
    printf("%d %d\n",a,b);
    if(a>b) return c;
    return mystery(a*2,b-1,c+1);
}
int main( void )
{
    int n = mystery(1,20,0);
    printf("%d\n",n);
    return 0;
}
```
For each of the following questions, write a function that computes the desired result. A good answer can fit in the available space, but use the back of the page if necessary.

Write a function that returns the number of vowels found in a string parameter. (5 points)

Write a function that compares two strings for near-equality. If the strings are the same length and differ by two or fewer letters, return true. Otherwise, return false. For example, comparing “frog” and “flow” would return true. (5 points)
Write a **program** that does the following: Roll two dice and add their values together. Repeat this process 1000 times. When done, open a file called “dice.txt” for writing, and write out the percentage of times each value was rolled. (10 points)

For example, dice.txt should contain “2 was rolled 2.7% of the time, 3 was rolled 5.8% of the time...”