1. Assume the definitions and initializations:

```
char c = 'T', d = 'S';
char *p1 = &c;
char *p2 = &d;
char *p3;
```

Assume further that the address of c is 6940, the address of d is 9772, and the address of e is 2224. What will be printed when the following statements are executed sequentially?

```
p3 = &d;
cout << "*p3 = " << *p3 << endl;    // (1)

p3 = p1;
cout << "*p3 = " << *p3             // (2)
     << ", p3 = " << p3 << endl;    // (3)

*p1 = *p2;
cout << "*p1 = " << *p1             // (4)
     << ", p1 = " << p1 << endl;    // (5)
```

2. Consider the following statements:

```
int *p;
int i;
int k;
i = 42;
k = i;
p = &i;
```

After these statements, which of the following statements will change the value of i to 75?

A. k = 75;
B. *k = 75;
C. p = 75;
D. *p = 75;
E. Two or more of the answers will change i to 75.

3. Explain the error.

```
char c = 'A';
double *p = &c;
```

4. Give the value of the left-hand side variable in each assignment statement. Assume the lines are executed sequentially. Assume the address of the blocks array is 4434.

```
int main()
{
   char blocks[3] = {'A','B','C'};
   char *ptr = &blocks[0];
   char temp;

   temp = blocks[0];
   temp = *(blocks + 2);
   temp = *(ptr + 1);
   temp = *ptr;

   ptr = blocks + 1;
   temp = *ptr;
   temp = *(ptr + 1);

   ptr = blocks;
   temp = *++ptr;
   temp = ++*ptr;
   temp = *ptr++;
   temp = *ptr;

   return 0;
```

For the following functions, **use the pointer notation ONLY**. Do NOT use the array index [] notation.

5. Write a piece of code which prints the characters in a cstring in a reverse order.

```
char s[10] = "abcde";
char* cptr;

// WRITE YOUR CODE HERE
```

6. Write a function countEven(int*, int) which receives an integer array and its size, and returns the number of even numbers in the array.

7. Write a function that returns a pointer to the maximum value of an array of double's. If the array is empty, return NULL.

```
double* maximum(double* a, int size);
```

8. Write a function myStrLen(char*) which returns the length of the parameter cstring. Write the function without using the C++ function strlen.

9. Write a function contains(char*, char) which returns true if the 1st parameter cstring contains the 2nd parameter char, or false otherwise.

10. Write a function revString(char*) which reverses the parameter cstring. The function returns nothing. You may use C++ string handling functions in <cstring> in the function if you wish.

```
int main()
{
  char s[10] = "abcde";
  revString(s);  // call the function
  return 0;
}

void revtString(char* ptr)
{
  // WRITE YOUR CODE HERE
}
```