CSC 3320 : System-Level Programming
Fall 2024: HW4 Key

1. (5 points) Ans: 24
   Explanation: `*(arr[1]+9)` means 9th element after the first element of the second row (pointed to by `arr[1]`), which is basically the last element of the 3rd row (i.e. 2, 4). Now consider the updated value: 10*2 + 4 = 24.

2. (5 points) Ans: Output: 12
   Explanation: The string "GeorgiaState" has 12 characters (G, e, o, r, g, i, a, S, t, a, t, e). The loop terminates when `str1` points to the null character. As `str2` was pointing to the beginning of the string, `(str1 - str2)` will be the length of the string, which is 12.

3. (5 points) Ans: "Computers Computers Computers "
   Explanation: This example is just another way of invoking a function using a pointer (to function). So, it is equivalent to the function call `fun(3)`, which prints "Computers " three times.

4. (5 points) Ans: The function can be re-written without using any `temp` variable as follows:

```
void swap(int *p, int *q)
{
   *p = *p + *q;
   *q = *p - *q;
   *p = *p - *q;
}
```

5. (5 points) Ans: A possible solution to the problem is as follows:

```
#include<stdio.h>

void find_two_largest(int a[], int n, int *largest, int *second_largest)
{
      int i;

      for(i=2; i<n; i++)
      {
          if(a[i] > *largest) {
              *second_largest = *largest;
              *largest = a[i];
          }
          else if(a[i] > *second_largest)
              *second_largest = a[i];
      }
      return ;
}

int main() {

   int a[10]={34, 23, 10, 56, 19, 29, 90, 12, 45};
   int largest, second_largest;

   if (a[0] > a[1]) {
      largest = a[0];
      second_largest = a[1];
   }
```

```
    else {
       largest = a[1];
       second_largest = a[0];
    }

    find_two_largest(a, 9, &largest, &second_largest);
    printf("Largest = %d\t Second Largest = %d", largest, second_largest);

    return 0;

}
// Largest = 90 Second Largest = 56
```

6. (5 points) Ans:

   Using single loop:

```
#define LEN 4
int sum_two_dimensional_array(const int a[] [LEN], int n)
{
    int *p, sum = 0;
    for (p = *a; p < *a + n*LEN; p++)
        sum += *p;
    return sum;
}
```

   Explanation: `*a` points to the first element of the first row and has type `(int *)`. So, if we just increase the pointer `p` (which was initially assigned to the first element of the first row), it will access elements in row-major order one-by-one.

7. (7 points) Ans: One of many possible solutions is as follows:

```
#include<stdio.h>
#include<stdbool.h>
#include<ctype.h>
#include<string.h>

bool isPalindrome(char* s) {

    bool flag = true;
    int i, j;
    i = 0; j= strlen(s) - 1;

    while(i<=j)
    {
        if(!isalnum(s[i]))
        {
            i++; continue;
        }
        else if(!isalnum(s[j]))
        {
            j--; continue;
        }
        else {
            if(toupper(s[i]) != toupper(s[j]))
            {
                flag = false; break;
```

```
            }
            else {
                i++; j--;
            }
        }
    }
    return flag;
}

int main() {

  printf("%s\n", isPalindrome("A man, a plan, a canal - Panama")? "true" : "false");
  return 0;
}
```

8. (5 points) Ans: 12 abc34 56 should be the answer.
   Explanation: `scanf` will first look for an integer. So, 12 will be assigned to `i`. Then it will look for a string and continue reading a string until a whitespace is found. So, `abc34` will be assigned to `s` (both alphabetic, numeric, and other characters can be part of a string). Next, it will look for another integer (skipping leading white spaces) and will assign 56 to `k`. The remaining part of the string will remain in the buffer as unread (potentially for subsequent `scanf` call).

9. (5 points) Ans: Output: Grinch
   Explanation: The program loops over a string `s` and subtracts one from each character to determine the immediate preceding character in the character set. So "H" will be replaced by "G", "s" by "r", and so on.

10. Ans:
    (a) (3 points) 3 - index of the first character (d) in "abcd" that doesn't appear in "babc".
    (b) (3 points (bonus)) 0 - index of the first character (a) in "abcd" that doesn't appear in "bcd".
    (c) (2 points (bonus)) It returns the index of the first character in `s` that does not appear in `t`. If all characters are found in `t`, then it simply returns the length of `s`.

Question:	1	2	3	4	5	6	7	8	9	10	Total
Points:	5	5	5	5	5	5	7	5	5	3	50
Bonus Points:	0	0	0	0	0	0	0	0	0	5	5
Score:											