CS 1713
Introduction to Computer Programming II
Midterm 2 Solutions

<table>
<thead>
<tr>
<th>Question</th>
<th>Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>Question 1</td>
<td></td>
</tr>
<tr>
<td>Question 2</td>
<td></td>
</tr>
<tr>
<td>Question 3</td>
<td></td>
</tr>
<tr>
<td>Question 4</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td></td>
</tr>
</tbody>
</table>

NAME:____________________________________

Instructions
1. Do all of the 4 problems
3. You have 50 minutes for the exam
4. Show all your work
5. Do not separate midterm papers

<table>
<thead>
<tr>
<th>Easy</th>
<th>Difficulty Level</th>
<th>Difficult</th>
</tr>
</thead>
<tbody>
<tr>
<td>□ □ □ □ □ □ □ □ □</td>
<td>□ □ □ □ □ □ □ □ □ □</td>
<td></td>
</tr>
<tr>
<td>1  2  3  4  5  6  7  8  9  10</td>
<td>1  2  3  4  5  6  7  8  9  10</td>
<td></td>
</tr>
</tbody>
</table>
1. (20 pts) What is the output of the following program? Show all your work.

```c
#include <stdio.h>

int mystery1(int *a, int b)
{
    *a = *a + b;
    return(*a+1);
}

int mystery2(int a, int *b)
{
    a = *b + 2;
    return(a+2);
}

int main()
{
    int n = 2, m = 5;
    int *nptr = &n;
    int *mptr = &m;

    printf("Output 1: n = %d, m = %d\n",n,m);
    n = mystery2(m,nptr);
    printf("Output 2: n = %d, m = %d\n",n,m);

    n = 2; m = 5;
    n = mystery1(mptr,n);
    printf("Output 3: n = %d, m = %d\n",n,m);

    n = 2; m = 5;
    n = mystery2(mystery1(nptr,1),mptr);
    printf("Output 4: n = %d, m = %d\n",n,m);

    n = 2; m = 5;
    n = mystery1(mptr,n)+mystery2(2,mptr);
    printf("Output 5: n = %d, m = %d\n",n,m);

    return 0;
}
```

Solution:

Output 1: n = 2, m = 5
Output 2: n = 6, m = 5
Output 3: n = 8, m = 7
Output 4: n = 9, m = 5
Output 5: n = 19, m = 7
2. (30 pts) Write a function *trimstring* to remove the spaces at the beginning of a string and return a pointer to the string. Do not allocate a new string. Update and return the original string. Sample executions of the function and the function prototype are given below.

trimstring(" An apple") returns the string "An apple"
trimstring(" Apple 2") returns the new string "Apple 2"

Complete the function below

**Solution**

```c
char *trimstring(char *str)
{
    int i=0;
    char *str1 = str;
    char *str2 = str;

    while (*str1 == ' ')
        str1++;

    while (*str1 != '\0')
    {
        *str = *str1;
        str++;
        str1++;
    }
    *str='\0';
    return(str2);
}
```
3. (20 pts) What is the output of the following program? Show all your work. Draw the contents of the array and pay attention to the formatting of the output.

```c
#include <stdio.h>
#include <stdlib.h>

void mystery(int *p, int n)
{
    int i;
    for (i=0; i<n; i++)
        printf("%d ",*(p+i));
    printf("\n");
    return;
}

int main()
{
    int i,*aptr,*aptr2;

    aptr = (int *)malloc(5*sizeof(int));

    for (i=0; i<5; i++)
        aptr[i]=(i+1)*(i+1);

    mystery(aptr,3);

    *aptr = 3;
    aptr2 = aptr;
    aptr2++;
    *aptr2 = 5;

    mystery(aptr+1,3);

    *(aptr+2) = 4;
    *(aptr2+2) = 2;

    mystery(aptr+2,3);
    free(aptr);
    return 0;
}
```

Solution:

1 4 9
5 9 16
4 2 25
4. (30 pts) Write a single function to find the pair of numbers closest to a provided number. Add the additional parameters you need for the pair of numbers into the function parameters. 

size is the size of array data and target is the number. Find the largest number smaller than target and the smallest number larger than the target. You can assume that all the array elements are distinct. You can assume that the number range from 0 to 1000.

For the following array and a target value of 9

```
12 7 4 13 8 5 6
```

Largest number smaller than target is 8 and smallest number larger than target is 12.

Add your code to the following template. You need to add two parameters for the two numbers.

Solution:

```c
void findclosestpair(int *data, int size, int target, int *low, int *high) {
    int i,j;
    *low = 1000;
    *high = 0;
    for (i=0; i<size; i++)
    {
        if (data[i]>target)
        {
            if (data[i]<*low)
            {
                *low = data[i];
            }
            else if (data[i]<target)
            {
                if (data[i]>*high)
                {
                    *high = data[i];
                }
            }
        }
    }
}
```