**CS 1713**  
Introduction to Computer Programming II  
Final 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>Question 5</td>
<td></td>
</tr>
<tr>
<td>Question 6</td>
<td></td>
</tr>
<tr>
<td>Question 7</td>
<td></td>
</tr>
<tr>
<td>Question 8</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td></td>
</tr>
</tbody>
</table>

**NAME:**

**Instructions**
1. Do **All** of the 8 problems
3. You have 120 minutes for the exam
4. Show all your work
5. Do not separate exam papers
1. (20 pts) What is the output of the following program? Show what is printed when `printlist()` functions are called. Show all your work.

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

struct node {
    int info;
    struct node *next;
};
typedef struct node node;

void mystery (node *head)
{
    if (head == NULL)
        return;
    while (head->next != NULL)
        head = head->next;
    printf("%d \n",head->info);
}

int main()
{
    node *head,*ptr,*ptr2;

    head = (node*)malloc(sizeof(node));
    head->next = (node*)malloc(sizeof(node));
    head->info = 12;
    head->next->info = 9;
    head->next->next = NULL;
    mystery(head);
    ptr = head->next;
    head->next = NULL;
    mystery(head);

    ptr2 = (node*)malloc(sizeof(node));
    ptr2 -> next = NULL;
    ptr2->info = 17;
    head -> next = ptr2;
    mystery(ptr2);
    ptr->info = 5;
    ptr2->next = ptr;
    mystery(head);
}
Solution:

9
12
17
5
2. (20 pts) Write a function to find a valley element in an array. A valley element is an element that is less than its neighbors. Given an input array where \( num[i] \neq num[i+1] \), find a valley element and return its index. The array may contain multiple valleys, in that case return the index to any one of the valleys. You may assume that \( num[1] = +\infty \) and \( num[n] = +\infty \). So, \( num[0] \) and \( num[n-1] \) could be peaks as well. For example, in array \([2, 1, 0, 1]\), 0 is a valley element and your function should return the index number 2.

In below function prototype, \( n \) is the size of the array

**Soluton:**

```c
int valley(int num[], int n)
{
    int i;
    int index = -1;

    if (num[0]<num[1])
        return (0);

    if (num[n-1]<num[n-2])
        return(n-1);

    for (i=1;i<n-1;i++)
        if (num[i]<num[i-1] && num[i]<num[i+1])
            {  
            index = i;
            return(i);
            }
    return(i);
}
```
3. (20 pts) Consider the following declaration of array of pointers

```c
int *data1[8];
int *data2[12];
```

Write a program fragment (for and if statements) to find out if there are two pointers in array `data1` and `data2` that point to the same location. If such pointers exist write the indexes of the arrays that point to the same location. Sample output is given below.

`data1[3] and data2[5] point to the same location`

**Solution:**

```c
for (i=0; i<8; i++)
    for (j=1; j<12; j++)
        if (data1[i] == data2[j])
            printf("data1[%d] and data2[%d] point to same place\n",i,j);
```
4. (20 pts) What is the output of the following program. Show all your work.

```c
#include <stdio.h>

struct triangle
{
    float width;
    float height;
};

typedef struct triangle triangle;

int main()
{
    triangle t1={3,4};
    triangle t2={1,2};
    triangle t3={4,6};
    triangle *tptr1=&t2;
    triangle *tptr2=&t1;

    printf("1 \%f \%f\n",t2.width, t2.height);
    printf("2 \%f \%f\n",tptr2->width, tptr2->height);

    tptr1 = &t3;
    printf("3 \%f \%f\n",(*tptr1).width, (*tptr1).height);

    tptr2->width = 5;
    tptr1 = tptr2;

    printf("4 \%f \%f\n",tptr1->width, tptr1->height);

    t1 = t2;
    printf("5 \%f \%f\n",t1.width, t1.height);
}
```

Solution:

1 1.000000 2.000000
2 3.000000 4.000000
3 4.000000 6.000000
4 5.000000 4.000000
5 1.000000 2.000000
5. (20 pts) Write a function to compute and return the count of most frequent character in a substring from positions i to j. If all the characters are different the function returns 1.

    highcount("apple",0,2) returns 2
    highcount("apple",0,1) returns 1

Prototype of the function is given below.

Solution:

```c
int highcount(char *str, int i, int j)
{
    int k,m,max,count;
    for (k=i; k<=j; k++)
    {
        count = 0;
        for (m=k+1; m<=j; m++)
            if (str[k]==str[m])
                count++;
        if (count > max)
            max = count;
    }
    return(max);
}
```
6. (20 pts) Span of a linked list is the difference between the largest and smallest elements in the linked list. Write a function `span()` to compute the span of a linked list. Node declaration of the linked list is given below.

```c
struct node
{
    int info;
    struct node *next;
};
typedef struct node node;
```

Span of the following list is 12-5=7.

```
head -> 5 -> 6 -> 12 -> Null
```

**Solution:**

```c
int span(node *ptr)
{
    int max, min;

    if (ptr == NULL)
        return(0);
    else
    {
        max=ptr->info;
        min=ptr->info;
    }

    while (ptr!=NULL)
    {
        if (ptr->info > max)
            max = ptr->info;
        if (ptr->info < min)
            min = ptr->info;
        ptr = ptr->next;
    }
    return(max-min);
}
```
#include <stdio.h>

int function1(int a, int b)
{
    if (a % b == 2)
        return a;
    else
        return (function1(a+b,a-b));
}

int main()
{
    int x,y;

    x = 7; y = 2;
    printf("Out1 = %d\n",function1(7,2));

    x = 8; y = 3;
    printf("Out1 = %d\n",function1(8,3));

    x = 11; y = 5;
    printf("Out1 = %d\n",function1(11,5));
}

Solution:

Out1 = 14
Out1 = 8
Out1 = 22
8. (20 pts) Write an function to find the lowest common ancestor in a binary search tree and return a pointer. Lowest common ancestor between two nodes \( a \) and \( b \) in tree \( T \) is defined as the lowest node in \( T \) that has both \( a \) and \( b \) as descendants. Node structure is given below.

```c
struct node {
    int key;
    struct node *left, *right;
};
typedef struct node node;
```

Consider the following tree as an example. lowest common ancestor of 4 and 8 is 6. lowest common ancestor of 4 and 14 is 10.

Solutions:

```c
node *lca(node *ptr, int a, int b)
{
    while (ptr != NULL)
    {
        if (ptr->key >= a && ptr->key <= b)
            return (ptr);
        else if (ptr->key > b)
            ptr = ptr->left;
        else if (ptr->key < a)
            ptr = ptr->right;
    }
    return (ptr);
}
```