<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>Total</td>
<td></td>
</tr>
</tbody>
</table>

NAME: ____________________________

**Instructions**
1. Do **ONLY** 6 of the 7 problems
3. You have 90 minutes for the exam
4. Show all your work
5. Do not separate exam papers

<table>
<thead>
<tr>
<th>Easy</th>
<th>Difficulty Level</th>
<th>Difficult</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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 printlist (node *head)
{
    while (head != NULL)
    {
        printf("%d ",head->info);
        head = head->next;
    }
    printf("\n");
}

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;
    printlist(head->next);

    ptr = head->next;
    printlist(head);

    ptr2 = (node*)malloc(sizeof(node));
    ptr2 ->next = ptr;
    ptr2->info = 17;
    head -> next = ptr2;
    printlist(ptr2);
    printlist(head);
}
```
2. (20 pts) Write a function to find a peak element in an array. A peak element is an element that is greater than its neighbors. Given an input array where $num[i] \neq num[i+1]$, find a peak element and return its index. The array may contain multiple peaks, in that case return the index to any one of the peaks. 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 $[1, 2, 3, 1]$, 3 is a peak element and your function should return the index number 2.

In below function prototype, $n$ is the size of the array

```c
typedef int peak(int num[], int n) {
    int i, 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 index;
}
```
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

```c
for (i=0; i<8; i++)
    for (j=0; j<12; j++)
        if (data1[i] == data2[j])
            printf("data1[%d] and data2[%d] point to the same location\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);
}
```

```
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 if a substring from positions i to j of a string consists of unique characters. If all the characters are different the function returns 1, otherwise it returns 0. Prototype of the function is given below.

```c
int isunique(char *str, int i, int j)
{
    int k, m;
    for (k = i; k <= j; k++)
        for (m = k + 1; m <= j; m++)
            if (str[k] == str[m])
                return (0);
    return(1);
}
```
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
```

```c
int span(node *ptr)
{
    int min, max
    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);
}
```
7. (20 pts) What is the output of the following program? Show all your work.

```c
#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));
}
```

```
Out1 = 7
Out1 = 14
Out1 = 22
```