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

NAME:_________________________

**Instructions**
1. Do all of the 6 problems
2. You have 90 minutes for the exam
3. Show all your work
4. 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>2</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td></td>
</tr>
</tbody>
</table>
1. (15 pts) What is the output of the following program? Show all your work.

```c
#include <stdio.h>

void function1(double *p1, double *p2)
{
    double *temp;
    temp = p1;
    p1 = p2;
    p2 = temp;
    return;
}

void function2(double *p1, double *p2)
{
    double temp;
    temp = *p1;
    *p1 = *p2;
    *p2 = temp;
    return;
}

int main()
{
    double x=2.3, y=3.4;
    double *ptr1=&x, *ptr2=&y;

    printf("Output1: %lf %lf\n", *ptr1, *ptr2);
    function1(ptr1, ptr2);
    printf("Output2: %lf %lf\n", *ptr1, *ptr2);

    ptr1 = &x;
    ptr2 = &y;
    function2(ptr1, ptr2);
    printf("Output3: %lf %lf\n", *ptr1, *ptr2);
}
```

Output1: 2.3 3.4
Output2: 2.3 3.4
Output3: 3.4 2.3
2. (25 pts) Consider the below definition of structure rectangle

```c
struct rectangle {
    float topleftx;
    float toplefty;
    float bottomrightx;
    float bottomrighty;
};
```

This structure represents a rectangle in cartesian coordinate system. Top left corner of the rectangle and bottom right corner of the rectangle determine the rectangle and other coordinates can be derived using these. `topleftx` and `toplefty` are the x and y coordinates of top left point and `bottomrightx` and `bottomrighty` are the x and y coordinates of the bottom right point of the rectangle.

Write a function `IntersectionRectangle` which takes as parameters two rectangles and returns -1 if the two rectangles do not intersect (overlap) and returns 1 of the two rectangles intersect. Write only the function.

```c
// Function to check intersection
int IntersectionRectangle(struct rectangle rect1, struct rectangle rect2) {
    // Implementation logic here
}
```
3. (20 pts) Consider the following declaration of array of pointers

```c
int *data[10];
```

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

6 and 8 point to the same location

```c
for (i=0; i<10; i++)
  for (j=i+1; j<10; j++)
    if (data[i][0] == data[j][0])
      printf("%d and %d point to the same location in i, j\n", i, j);
```
4. (15 pts) What is the output of the following program. Show all your work.

```
#include <stdio.h>

struct cylinder {
    float radius;
    float height;
};

typedef struct cylinder cylinder;

int main()
{
    cylinder c1={3,5};
    cylinder c2={2,4};
    cylinder *cptr1=&c2;
    cylinder *cptr2=&c1;

    printf("1 %f %f\n", c2.radius, c2.height);
    printf("2 %f %f\n", cptr2->radius, cptr2->height);
    printf("3 %f %f\n", (*cptr1).radius, (*cptr1).height);

    cptr1 = cptr2;
    printf("4 %f %f\n", cptr2->radius, cptr2->height);

    c1 = c2;
    printf("5 %f %f\n", c1.radius, c1.height);
}
```
5. (25 pts) Complete the following function to compute the largest palindrome in str. str has many palindromes in it including ABCBA, RACECAR, ARA, IAMA. Largest palindrome is NEVERODDOREVEN since it has the most characters in it. Store the largest palindrome in string pal. You can assume that a function

```c
int palindromeLength(char *str, int i, int j)
```

is available for you to use. This function takes a string str and two points i and j and determines whether the string from i to j is a palindrome. If it is a palindrome, it returns the length of the palindrome and if it is not a palindrome, it returns -1.

```c
int main()
{
    int i, j;
    char str[100];
    char pal[100];
    int max = 0;
    strcpy(str,"ABCBAHELLOHOWRACECARAREYOUIAMAOOINEVERODDOREVENNGGOOD");
    for (i = 0; i < strlen(str); i++)
        for (j = i + 1; j < strlen(str); j++)
            if (palindromeLength(str, i, j) > max)
                max = j - i + 1;

    for (k = 0; k < j - i + 1; k++)
        pal[k] = str[i+k];

    printf("Largest Palindrome = %s\n", pal);
}
```
6. (20 pts) Write a program that creates a table that shows the down payments for a house purchase if the down payment rate is 10%, 20% and 30%. Write the table to file `mortgage.txt`. Each line of the file contains the price, and 10%, 20% and 30% down payments respectively for that price. In the table to be created, price varies from 100,000 to 300,000 in increments of 10,000. First few lines and last two lines of the file is as follows.

<table>
<thead>
<tr>
<th>Price</th>
<th>10 perc.</th>
<th>20 perc.</th>
<th>30 perc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>100000</td>
<td>10000</td>
<td>20000</td>
<td>30000</td>
</tr>
<tr>
<td>110000</td>
<td>11000</td>
<td>22000</td>
<td>33000</td>
</tr>
<tr>
<td>120000</td>
<td>12000</td>
<td>24000</td>
<td>36000</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>290000</td>
<td>29000</td>
<td>58000</td>
<td>87000</td>
</tr>
<tr>
<td>300000</td>
<td>30000</td>
<td>60000</td>
<td>90000</td>
</tr>
</tbody>
</table>

```c
#include <stdio.h>

int main()
{
    FILE *pfile;
    float i = 0;
    pfile = fopen("mortgage.txt", "w");
    fprintf(pfile, "Price 10 perc. 20 perc. 30 perc. 
");
    for (i = 100000; i <= 300000; i = i + 10000)
    {
        fprintf(pfile, "%10.0f %10.0f %10.0f %10.0f 
", i, i*0.1, i*0.2, i*0.3);
    }
    fclose(pfile);
}
```