Homework Assignment 9

Q1. Consider the following declaration of a 'two-dimensional array in C:

```
char a[100][100];
```

Assuming that the main memory is byte-addressable and that the array is stored starting from memory address 0, the address of a[40][50] is

Q2. The value of a[2] after executing the code below is:

```
int a[5] = {0};
for (i = 1; i < 5; i++)
a[i] = a[i -1] + i;
```

Q3. What is the output ?

```
int foo6 ( char A[] , unsigned int n )
    {
       int t;

       if (n == 0) return 0;
       t = foo6(A,n-1);
       if ( ((A[n-1]>='a') && (A[n-1]<='z')) ||
            ((A[n-1]>='A') && (A[n-1]<='Z')) ||
            ((A[n-1]>='0') && (A[n-1]<='9')) )
          ++t;
       return t;
    }
```

Q4. What is the output ?

```
#include <stdio.h>
int main()
{
     char arr[] = "ProgCQuiz";
     printf("%s", (arr+5));
     return 0;
}
```

Q5. Consider the following C-function in which a[n] and b[m] are two sorted integer arrays and c[n + m] be another integer array.

```
void xyz(int a[], int b [], int c[])
{
int i, j, k;
i = j = k = 0;
while ((i<n) && (j<m))
     if (a[i] < b[j]) c[k++] = a[i++];
     else c[k++] = b[j++];
}
```

Which of the following condition(s) hold(s) after the termination of the while loop?

(i) j < m, k = n+j-1, and a[n-1] < b[j] if i = n

(ii) i < n, k = m+i-1, and b[m-1] <= a[i] if j = m

(A) only (i) **(B)** only (ii) **(C)** either (i) or (ii) but not both
(D) neither (i) nor (ii)

Q5. What will be the output

```
# include <stdio.h>

void print(int arr[])
{
int n = sizeof(arr)/sizeof(arr[0]);
int i;
for (i = 0; i < n; i++)
     printf("%d ", arr[i]);
}

int main()
{
int arr[] = {1, 2, 3, 4, 5, 6, 7, 8};
print(arr);
return 0;
}
```

Q6. Consider the following way to compute the maximum in an array A of size n. First, divide the array in two equal (or almost equal) halves. Then, recursively compute the maximums in the two halves. Finally, return the larger of these two recursively computed maximum values. Complete the following function which uses the above idea for finding the maximum in an array.

```
int max ( --------, int n )
{
    int m1, m2;
    if (n == 1) return——---;

    /*Make two recursive calls. Do not assume that n is even.*/

   m1 = max(----,-----);
  m2 = max(----,-----);
  return(-------------);
}
```

Q7. (a) Given a polynomial of degree n, $p(x) = a_n x^n + a_{n-1} x^{n-1} + ...a_1 x^1 + a_0$, $a_n = 0$, it can be rewritten as: $p(x) = (((a_n x + a_{n-1})x + ...a_1)x + a_0$. This is the Horner's scheme for evaluating p(x). The advantage of this method of evaluation is that explicit exponentiation is avoided. Let the coefficients for the various powers of x of p(x) be stored in an array (say) **P[ ], as float P[ ]=$\{a_n, a_{n-1}, ..., a_1, a_0\}$.**
Write an iterative and recursive function to compute p(x)
