CLASS TEST – 2					**SEMESTER: Spring 2016**				
Roll Number					**Name**				
PDS Section					**Subject Name**	***Programming & Data Structures***			
Department/Centre of the Student									
To be filled in by the Examiner									
Question Number	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**Total**
Marks obtained									

Date: April 07, 2016 **Marks: 50** **Time: 1 hour**

Answer all the questions. Write answers within the space provided.

Q1. What is printed by the program below? **[6 marks]**

```
#include <stdio.h>
int scan (int *a, int m)
{
  static rev=0;
  int j,k;
  for (j=0;j<m-1;j++)
    if (a[j]>a[j+1]) {rev++; k=a[j]; a[j]=a[j+1]; a[j+1]=k;};
  return(rev);
}
int main ()
{
  int i, n=6;
  int a[20] = { 6, 5, 4, 3, 2, 1 };
  for (i=0;i<n-1;i++)  printf("%d ", scan(a,n-i));
  printf("\n");
}
```

[Answer: 5 9 12 14 15]

Q2. Fill in the five blanks suitably so that the function **binary_search** searches array **A** between (and including) indices **imin** and **imax**, i.e. for array elements **A[imin]** through **A[imax]**, where **imax≥imin**. If the search finds the integer **key** amongst these elements then the function returns an index **i** in **A** where **key** is found (that is **A[i]=key**); otherwise, it returns **-1**. **[5 marks]**

```
int binary_search(int A[], int key, int imin, int imax)
{
  int imid;
   if (imax < _____________________)
    return -1;
  else
    {
      imid = (_____________________)/2;
      if (A[imid] > key)
         return binary_search(A, key,___________,imid-1);
      else if (A[imid] < key)
         return binary_search(A, key, ____________,imax);
      else
         return _________________;
    }
}
```

Answer:
imin
imin+imax
imin
imid+1
imid

Q3. Given the following declarations: **[6 marks]**

```
int i,j;
int a[][3]={{1,2,3},{4,5,6},{7,8,9},{10,11,12}};
int b[3][4];
```

what does the following code print?

```
for (i=0;i<4;i++) {
  for (j=0;j<3;j++)
     *(b[j]+i) = *(*(a+i)+j);
}

for (i=0;i<3;i++){
  for (j=0;j<4;j++)
printf ("%d ", b[i][j]);  printf("\n");
}
```

Answer:
1 4 7 10
2 5 8 11
3 6 9 12

Q4. What is printed by the following program? **[6 marks]**

```
#include <stdio.h>
#include <malloc.h>
struct student
{
char name[10]; int x;
};

int main()
{
  int n,i,j;
  float sum;
  struct student list[50];
  *(list->name)='f'; *(list->name+1)='1'; *(list->name+2)='\0';
  *((list+1)->name)='a'; *( (list+1)->name+1)='a';
  *((list+1)->name+2)='\0'; (list+1)->x=14;
  *(list+2)=*(list+1);
  printf("%s\n",list->name); printf("%s\n", list[2].name);
  printf("%d\n",(list+2)->x); printf("%s\n",(list+1)->name);
}
```

Answer:
f1
aa
14
aa

Q5. Fill in the blanks in such a way that the following program prints: **a[0]=50, a[1]=10**. **[5 marks]**

```
int *foo( )
{
   int x=10, y=50;
   int *z;

    _____________________________________________;  // Allocate space

   *z = _________________;   *(_________________) = x;
   return z; }

void main( )
{
   int *a;
   a = foo();
   printf("a[0] = %d, a[1] = %d\n", a[0], a[1]);
}
```

Ans:
z = malloc(2*sizeof(int));
*z = y OR *z = 50;
*(z+1) = x;

Q6. Complete the structure definition and initialization of **emp**, such that the program prints:

```
Ramesh 1000 1000.5 22,6,1990
```

[6 marks]

```
#include <stdio.h>
struct Employee
{
    char ename[20];
    int ssn;
    float salary;

    _____;

} emp = {_____ };

int main(int argc, char *argv[])
{
  printf("%s ",emp.ename);    printf("%d ",emp.ssn);
  printf("%f ",emp.salary);
  printf(" %d,%d,%d",  emp.doj.date, emp.doj.month, emp.doj.year);
  return 0;
}
```

Answer:

```
struct { int date; int month; int year; }
   doj;

"Ramesh", 1000, 1000.50, {22,6,1990}
```

Q7. What will be printed by the following C code segments? **[3+3=6 marks]**

```
int c[20] = {0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19},
    (*d)[4] = (int (*)[4])(c+5);
printf ("d[2][3]: %d\n", d[2][3]);
```

```
d[2][3]: 16
```

```
int e[3][5] = {{1,2,3,4,5}, {-1,-2,-3,-4,-5}, {6,7,8,9,0}},
*f = e[1]+2;
printf ("f[5]: %d\n", f[5]);
```

```
f[5]: 8
```

Q8. A string of characters *x* is a ***suffix*** of another string *y*, if *x* is same as the ***tail*** of *y*. As an example "IIT" is a suffix of "Kharagpur IIT", but "IIIT" is not a suffix of "Kharagpur IIT". "Kharagpur IIT" is its own suffix. A NULL string is a suffix of any string. The following recursive function takes two strings, *x*, *y*, and their lengths as parameters. It returns 1 if *x* is a suffix of *y*, otherwise it returns 0.

```
int isSuffix( char *x, char *y, int lenX, int lenY)
{
  if (β1) return 1;
  if (β2) return 0;
  if (β3) return isSuffix (x, y, β4, β5);
  return 0;
}
```

What are the expressions for β_1, β_2, β_3, β_4, and β_5? **[5×2=10 Marks]**

$\beta_{1:}$ `lenX < 0`	β_2: `lenX > lenY`
β_3: `x[lenX] == y[lenY]`	β_4: `lenX - 1`
β_5: `lenY - 1`	