Indian Institute of Technology Kharagpur
Department of Computer Science & Engineering

**Programming & Data Structures (CS11001/CS13002)** **Autumn Semester 2009**
**Max. Time: 1 Hour** **Max. Marks: 50**

**Instructions**:

I. Answer all questions.
II. Answer on the question paper itself. No separate answer sheet will be given. Do any rough work on the blank page at the end of the question paper.

---

Q. 1 What will be the output of the following program? Justify your answer. **[4]**

```
#include <stdio.h>
main(){
int i=4, a[5]={1,2,3,4,5};

while(*(a+i)>=i){
        i=i/2;
        printf("a[i]=%d i= %d\n",a[i],i);}
}
```

**Ans**

3 2 1 Mark
2 1 1 Mark
1 0 1 Mark
1 0
… to be continued in an infinite loop. 1 Mark

Q.2 What will be the output of the following program? If you feel that there can be any compile time or run time error, point out the same. Justify your answer. **[1+1]**

```
float *fp;
*fp=2.134;
print("%f\n",*fp);
```

**Ans**

**Runtime error will occur.** 1 Mark
**Storage for *fp has not been allocated.** 1 Mark

Q. 3 What will be the output of the following programs? If you feel that there can be any compile time or run time error, point out the same. Justify your answer. **[1+1]**

```
#include <stdio.h>
main(){
  int i,a[10];
  int *j;
```

```
    for(i=0;i<10;i++)a[i]=i*2;
    j=a;
    for(i=0;i<10;i++)  printf("%d\n", (*j)++);
}
```

**Ans** **0 1 2 3 4 5 6 7 8 9** **1 Mark**

***j is a[0] which is 0. (*j)++ increments a[0] value each time. 1 Mark**

Q.4 Consider the following code. Write the missing parts of the code as indicated by the comment statements. **[1+1]**

```
int i=10,*j,**k;
j=&i;
.....                                   /* Write a statement to print the value of i using j*/
k=&j;
.....                                   /* Write a statement to print the value of i using k*/
```

**Ans** **printf("%d\n",*j);** **1 Mark**

**printf("%d\n",**k);** **1 Mark**

Q.5. What will be the output of the following programs? If you feel that there can be any compile time or run time error, point out the same. Justify your answer. If there is any error, correct the same. **[1+1+1+1]**

```
#include stdio.h
main(){
        int a[5]={1,2,3,4,5};
        for(i=0;i<5;i++) printf("%d\n", *a++);}
```

**Ans**

**Compile time error will occur.** **1 Mark**

**<stdio.h> in place of stdio.h** **1 Mark**

***a++ is an attempt to increment a constant pointer.** **1 Mark**

**The for loop should be replaced by**

**for(i=0;i<5;i++) printf("%d\n", *(a+i));** **1 Mark**

**Marking guideline: Answers such as *a+i is incorrect. There are other variations. Please consider e.g. *(a+i) may be a[i]. No part marking.**

Q.6. Consider the following code: **[5]**

```
typedef  struct _Abc{
        int emp_id;
        char name[20];
        char addr[25];
        double salary;
      } Abc;

      Abc *abc;
```

Assign a valid address to the pointer variable "abc" in the following two ways: (i) using an existing address of an appropriate variable, and (ii) by dynamically allocating the appropriate memory. Write code for printing all the data-fields which are pointed to by the pointer variable "abc".

**Ans.:** **Award:**

```
// for part (i)
Abc a;     OR      Abc a = {1234, "RKumar", "CS112", 567.88};     -- 1 mark

abc = &a;                                                          -- 1 mark

// For  Part (ii)
abc = (Abc *) malloc(sizeof(Abc));                                  -- 1 mark

// For printing all the data-fields
printf("{ %d, %s, %s, %lf}\n",abc->emp_id, abc->name, abc->addr, abc->salary);
OR                                                                 -- 2 marks
printf("{ %d, %s, %s, %lf}\n",(*abc).emp_id, (*abc).name, (*abc).addr, (*abc).salary);
```

**Marking Guidelines**: One may use **struct _Abc** in place of **Abc**. However **struct Abc** is not valid. No part marking for any error.

~~Q.7. What will be the output of the following program? If you feel that there is any compile-time or run-time error, point out the same and justify your reasoning. In case there is any error, correct the errors. Show the output that would be generated.~~ **~~[1+1+2]~~**

```
#include stdio.h
Main(){
   int a[5]={1,2,3,4,5};
   for(i=0;i<5;i++) printf("%d\n",*a++);
}
```

Q.8. Identify the valid and invalid assignment statements in the following piece of the code segment. Briefly write complete and proper reasoning for each of your answer. **[5]**

```
double * abc, *bcd;
double dA, dB, dArr[10] ;
int  iArr[10] ;

&dB = &dA ;   //invalid
bcd = dArr ;          ok
dArr = (double *)iArr ;  //invalid
abc = (double *)iArr ;  ok
abc=bcd;  ok
```

-- No Award for simply writing Valid/Invalid.
-- Award 1 mark for each correct part. You may award ½ Mark not stating the Correct and Complete reason, as stated below.

**Ans.:**

**&dB = &dA;  INVALID -- &dB holds a const address which cannot be an LValue.**

**bcd = dArr;  VALID -- bcd gets the address of the first element of dArr;**
**-- both have type (double *)**

**dArr = (double*)iArr;  INVALID -- Though both have the same type (double *),**
**dArr holds a const address which cannot be an LValue.**

**abc = (double *)iArr ;  VALID -- abc gets the address of first element of iArr**
**-- Type casting results in Type Match.**

**abc = bcd;  VALID -- Though bcd does not point to a valid address, yet assignment is OK.**

Q. 9. Fill in the blanks for performing operations as indicated in the comments. Write the output. **[2+1+1+1]**

```
char *p;

p= (char *) malloc(20*sizeof(char)) ;/*Dynamically allocate an array of 20 characters or
                                    (char *)  calloc(20, sizeof(char)) */   2 Marks (Deduct 1 Mark if
                                                                            type casting is not made.)

if (p==NULL) { /* Check for proper allocation */    1 Mark (No part marking.)
      printf("Error in memory allocation. \n");
      exit(0);
 }
strcpy(p,"IITKGP");  1 Mark  (No part marking.)
```

```
                                    /* Copy a string "IITKGP" to the allocated array using strcpy() function. */
   printf("%s \n", p+3);
```

The output: KGP 1 *Mark* (No part marking.)

Q10. The following incomplete code is for a **recursive function** RecursiveArraySum, which calculates the sum of all the n-elements of an array passed as function argument. Write the missing pieces of the code such that the sum is calculated by recursively calling the function. The function RecursiveArraySum is invoked with a 4-element array of [5, 10, 20, 40] as argument. The base address of the array is at 10,000 (decimal). At every recursive call, write the values of the parameters passed and the return value from the function.
**[7]**

```
int RecursiveArraySum(int *arr, int n){

 if   . . . . . . . . return . . . . . ;

    ……..

    …… RecursiveArraySum(……..……) ……..
}
```

**Ans.:** **Award:**

```
if (n == 0)  return 0;                          -- 0.5 + 0.5  mark
  else
 return (a[0] + RecursiveArraySum(a+1,n-1));    -- 1 + 1 + 1 marks

OR

return (a[n-1] + RecursiveArraySum(a,n-1));
```

**Calls to Recursive Functions: The following corresponds to**
**`return (a[0] + RecursiveArraySum(a+1,n-1));`**

| Address (a) | n | Return value |
|---|---|---|
| 10000 | 4 | 5 + . . . |
| 10004 | 3 | 10 + |
| 10008 | 2 | 20 + |
| 10012 | 1 | 40 + |
| 10016 | 0 | 0 |

OR

For the second type of recursive call:

| Address (a) | n | Return value |
|---|---|---|
| 10000 | 4 | 40 + . . . |
| 10000 | 3 | 20 + |
| 10000 | 2 | 10 + |

**10000 1 5 +**
**10000 0 0**

**Award : 3 marks in total ( 1 Mark each for correct values of addresses, n and Return Value of any of the two possiblities).**

Q. 11. Consider a computer that has 4 byte memory addresses. What will be the output of the following statement? Justify your answer. **[2]**

```
printf("%d %d %d\n", sizeof(char *), sizeof(int *), sizeof(float *));
```

**Ans** **4 4 4** **1 Mark**

**Each pointer stores an address of a memory location which is given to be 4 bytes. 1 Mark**

Q. 12. Correct the following code by only adding/deleting operators without any overwriting. Give reasoning behind your code changes. Write the output thereafter. **[4]**

```
typedef struct {
        float real;
        float imag;
    } _COMPLEX;

main( ) {
 _COMPLEX a = {3.0,  -4.0},  *p;

    p = & a; /* Assign the address of a not its value. */   Award 1 Marks.
    printf("Values: %f %f \n",  (*p) . real,  (*p) . imag); /*' . ' is a higher precedence operator. For
                                    referencing to pointer brackets should be used. */ Award 1 Marks.

    p->real=p->imag=2.5;
    printf("%f %f \n", p->real, p->imag); /* Use of appropriate operator ('->') for accessing the member
                                    variables of the structure pointed by p. One may also write
                                    them as (*p).real and (*p).imag. */   Award 1 Marks.

}
```

Output: Award 1 Marks.

3.0 -4.0
2.5 2.5

Explanations are stated within the comments after changes.

Marking Guidelines: Changes may be made in different ways (e.g. in stead of **(*p).real** it may be **p->real** etc.) for accessing member variables using a pointer. Without any explanation, no marks would be awarded. However explanation may be provided in different formats and for any reasonable answer award full marks to each changes. Deduct ½ for each wrong updates.

Q. 13. Change the following code so that whenever a call to the update() function is made, it updates the variable **A** of **main( )**). You can insert additional referencing (‘**&**’) and dereferencing (‘*****’) operators for modifying the code. Do not delete any part of the cods. Briefly justify each change made. What would be the output from the modified code? **[4+2]**

```
#include <stdio.h>
int update(int    *acc,  int    x) {  /* To retain changes in a variable  pointer variables are used and
                                        initialized with the address of the variable passed as an argument. */
                                                                                    Award 1 Marks.

    *acc   =    *acc +   x; /* Accessing acc with dereferencing. */              Award 1 Marks.

    if (  *acc  >  0 )   return   1; /* Accessing acc with dereferencing. */      Award 1 Marks.

    else    return    0;
}

main( )   {
   int A  =  0,   s;

   s  =  update(   & A ,   10 ); /* Pass address of A */            Award 1 Marks.

   s  =  update(   &  A ,   -9 ); /* Pass address of A */           Award 1 Marks.

   printf("%d %d \n",   A,    s);
}
```

Output: 1 1 Award 1 Marks.

**Marking Guidelines: Without any explanation, no marks would be awarded. However explanation may be provided in different formats and for any reasonable answer award full marks to each changes. Deduct ½ for each wrong updates.**