DR. BABASAHEB AMBEDKAR TECHNOLOGICAL UNIVERSITY
LONERE – RAIGAD – 402 103
Semester Examination – December – 2017

Branch: B. Tech.(Group A / Group B) **Semester: I**

Subject with Subject Code: Basic Computer Programming **Marks: 60**

[ICT106] MODEL ANSWER

Date: 20 / 12 / 2017 **Time: 3 Hrs.**

==========

Q. No. 1 **[4]**

A)i) Answer: The curly braces denote a block of code, in which variables can be declared. Variables declared within the block are valid only until the end of the block, marked by the matching right curly brace '}'. The body of a function is one such type of block, and thus, curly braces are used to describe the extent of that block. **[2]**

ii) `double ans = 10.0+2.0/((3.0−2.0)∗2.0);` **[1]**

iii) Preprocessor macros like `#include` should not be terminated with semicolons. The correct code is:

```
#include <stdio.h>
```

B) **[4]**

Creating Source Code	**1 Mark**
Compile Source Code	**1 Mark**
Executing / Running Executable File	**1 Mark**
Flowchart	**1 Mark**

C) **[4]**

```
#include <stdio.h>
void main(){
    int a, b, c, m;
    printf("Enter a THREE integers: ");
    scanf("%d %d %d", &a, &b, &c);
    if(a>b){
      m = a;
    }
    else{
      m = b;
    }
    if(m < c){
      m = c;
    }
    printf("Maximum value among THREE = %d", &m);
}
```

1 Mark **1 Mark** **1 Mark** **1 Mark**

Q. No. 2

A) Each carry **1 Mark** **[4]**

```
i. c>='a' && c<='z'
ii.      c>='A' && c<='Z'
iii.     c>='0' && c<='9'
iv.      c =='\n' || c<='\t' || c == ' '
```

B) (1 && ((0 % 10) >= 0)) && ((30 % 10) <= 3) **ANS: TRUE** **[4]**

Priority:

i. modulus
ii. relational operators less than and grater than
iii. Logical AND

For Correct ANSWER and with proper PRESECEDENCE TWO Marks.

Write the values of the following expressions.

i. `'F' - 'C'` **ANS: 3** **1 Mark for correct Answer**

ii. `2.0 + (float)(5/3)` **ANS: 3.0** **1 Mark for correct Answer**

OR

B) For each operator give proper example and associativity of each operator **CARRY ONE MARK EACH.** **[4]**

C) In C variable is a named location in a memory where a program can manipulate the data. This location is used to hold the value of the variable. **1 Mark** **[4]**

Rules for naming C variable: **1 Mark**

1. Variable name must begin with letter or underscore.
2. Variables are case sensitive
3. They can be constructed with digits, letters.
4. No special symbols are allowed other than underscore.
5. sum, height, _value are some examples for variable name

Variable Declaration: **1 Mark**

data_type variable_name;

Example: int x, y, z; char flat, ch;

Variable initialization: **1 Mark**

data_type variable_name = value;

Example: int x = 50, y = 30; char flag = ‘x’, ch=’l’;

Q. No. 3

A) **[4]**

```c
#include<stdio.h>
#include<conio.h>
void main()
{
int n, p;
float amount;
clrscr();
printf("Enter the customer number: ");
scanf("%d",&n);
printf("Enter the No. of Calls: ");
scanf("%d",&p);                          //for both the input
 if(p>=0 && p<=200)
    amount=p*0.50;
 else if(p>200 && p<400)
    amount = 100+((p-200) * 0.65);
 else if(p>400 && p<=600)
    amount = 230 + ((p-400) * 0.80);
printf("Amount to be paid by customer no. %d is Rs.:5.2f.",n,amount);
getch();
}
```

1 Mark
1 Mark
1 Mark
1 Mark

OR

A) **[4]**

```c
#include <stdio.h>
int main(){
long int n;
int i, d;
int s=0;
int sum=0;
scanf("%ld", &n);
while(n){                        // first loop
      d = n%10;
      s = s+d;
      n = n/10;
}
while(s){                         // second loop
      i = s%10;
      sum = sum + i;
      s = s/10;
}
printf("%d", sum);
}
```

B) **[4]**

For Both the Sub-question:
Write syntax of each **1 mark**
Basic Difference **1 mark**

C) [4]

i. switch(expression) {

```
   case constant-expression  :
      statement(s);
      break; /* optional */

   case constant-expression  :
      statement(s);
      break; /* optional */

   /* you can have any number of case statements */
   default : /* Optional */
   statement(s);
}
```

1 Mark for Syntax

ii. 1: 11 12 13 14 15 16 17 18 19
2: 22 23 24 25 26 27 28 29
3:
4: 44 45 46 47 48 49
5: 55 56 57 58 59
6:
7: 77 78 79
8: 88 89
9:

3 Marks for output

Q. No. 4

A) [3]

```
5,5,6
```

1 Mark for each correct value

B) [3]

```
#include <stdio.h>
void armstrong(int n);

void main(){
    int number;
    printf("Enter a three digit integer: ");
    scanf("%d", &number);
    armstrong(number)
}

void armstrong(int n){
    int originalNumber, remainder, result = 0;
    originalNumber = n;
    while (originalNumber!= 0){
        remainder = originalNumber%10;
        result += remainder*remainder*remainder;
        originalNumber /= 10;
    }

    if(result == n)
        printf("Enter Number is %d an Armstrong number.",n);
    else
        printf(" Enter Number is %d not an Armstrong number.",n);

    return 0;
}
```

Main Function 1 Mark

Logic of Armstrong Number Function 1 Mark

if else part 1 Mark

C) **[6]**

i. When Function is call within same function is called **Recursion**. The function which call same function is called **recursive function**. In other word when a function call itself then that function is called **Recursive function**.

e.g.
```
void recurse()
{
    recurse(); /* Function calls itself */
}

int main()
{
    recurse(); /* Sets off the recursion */
    return 0;
}
```

1 Mark for definition and example.

ii.

Function Definition: **1 Mark**

```
Return_type function_name(arguments list){
      Body of function;
}
```

Function Call: **1 Mark**

```
function_name (arguments list);
```

Function Prototype: **1 Mark**

```
This informs compiler about the function name, function parameters and
return value's data type.

return_type function_name (argument list);
```

iii. 1 Mark for explanation of any storage class and 1 Mark for program.

Q. No. 5

A) **[4]**

```
#include <stdio.h>

int main()
{
    int n, i;
    float num[100], sum = 0.0, average;                    1 Mark

    printf("Enter the numbers of elements: ");
    scanf("%d", &n);                                       1 Mark

    while (n > 100 || n <= 0){    //optional
        printf("Error! number should in range of (1 to 100).\n");
        printf("Enter the number again: ");
        scanf("%d", &n);
    }

    for(i = 0; i < n; ++i)  {                              1 Mark
        printf("%d. Enter number: ", i+1);
        scanf("%f", &num[i]);
```

```
        sum += num[i];
    }

    average = sum / n;
    printf("Average = %.2f", average);

    return 0;
}
```

1 Mark

B) **[3]**

```
f[0]=27
f[2]=73
f[3]=73
f[2]=65
f[4]=73
f[5]=73
f[4]=68
f[3]=65
```

at least 6 value of f[]. Each carry half mark.

C) **[5]**

An **array** is a collection of data items, all of the same type, accessed using a common name. **1 Mark**

int a[3][4] = {0,1,2,3,4,5,6,7,8,9,10,11}; **1 Mark**

	0	1	2	3
a [3][4] =	4	5	6	7
	8	9	10	11

1 Mark

Write C statements for one dimension array to do the following:

i. a[5] = 35; **1 Mark**

ii. a[9] = a[6] + a[13]; **1 Mark**

Q. No. 6

A) **[3]**

```
struct Student{
    char name[25];
    int age;
    char gender;   //F for female and M for male
    char department[20];
};
```

2 Marks

`struct Student stud1 = {xyz, 23, Male, Computer };` **1 Mark**

B) **[6]**

```
#include<stdio.h>
#include<conio.h>
struct emp
{
    int empno;
    char name[10];
    char dept[30];
```

```
    char address[100];
    int salary;
} e[5];
```
2 Marks

```
void main()
{
    int i;
    clrscr();
    for(i = 0; i < 5; i++)
    {
        printf("\nEnter the employee number: ");
        scanf("%d", &e[i].empno);
        printf("\nEnter the name: ");
        scanf("%s", e[i].name);
        printf("\nEnter the employee's Department: ");
        scanf("%s", &e[i].dept);
        printf("\nEnter the employee's Address: ");
        scanf("%5", &e[i].address);
        printf("\nEnter the salary: ");
        scanf("%d", &e[i].salary);
```
2 Marks

```
    }
    for(i = 0; i < 5; i++)
    {
       printf("%d \t %s \t %s \t %s \t %d\n", e[i].empno, e[i].name,
        e[i].dept, e[i].address, e[i].salary);
    }
```
2 Marks

```
    getch();
}
```

C) **[3]**

```
struct Date{
      int month;
      int day;
      int year;
};
```
1 Mark

```
struct Account{
      int acc_no;
      char acc_type;          // 'S' for Saving and 'C' for Current
      char name[30];
      float balance;
      struct Date lastpayment;
};
```
1 Mark

to initialization of all members of both structures using dot operator. **1 Mark**