

**FIRST SEMESTER EXAMINATION, 2009-2010****COMPUTER CONCEPTS AND PROGRAMMING IN C***Time : 3 Hours**Total Marks : 100*

- Note :** 1. This question paper consists of three sections. Section A contains objective type questions and is of 20 marks. Section B consists of short answer type questions which of 30 marks and Section C contains long answer type questions of total 50 marks.
2. Your answers for Section B and C should be precise and to the point.
3. Answer to the questions of each section must be done at one place in your answer books.
4. You are required to attempt all the questions.

**SECTION - A**

1. There are total 20 multiple choice questions. Only one of the answer out of given four

Choices is correct. Write the correct answer. (10 × 1 = 10)

- (i) In evaluation is correct
- (a) + has higher precedence over\*
- (b) \* has higher precedence over +
- (c) both \* and + have the same precedence
- (d) The order of evaluation does not matter

Ans. (b) \* has higher precedence over +

- (ii) A Stack is

- (a) LIFO (Last in First out)
- (b) FIFO (First in First out)
- (c) LILO (Las + Las + out)
- (d) None of the above

Ans. (a) LIFO (Last in First out)

- (iii) An array can store

- (a) Finite data of similar type

- (b) Infinite data of similar type
- (c) Finite data of mix type
- (d) All of the above

Ans. (a) Finite data of similar type

- (iv) How many bytes of storage an unsigned short integer in C language would require

- (a) 2
- (b) 4
- (c) 6
- (d) 8

Ans. (c) 6

- (v) Decimal number 10 can be represented in unary (a number system with base 1) as

- (a) 1010
- (b) 64
- (c) A
- (d) None of the above

Ans. (d) None of the above

- (vi) If  $k = 5$  then the value of variable  $x$  after the execution of a C statement  $x = k++$  will be

- (a) 5
- (b) 6
- (c) randomly any one of the above
- (d) value of x will not depend on k

Ans. (d) value of x will not depend on k

(vii) Typically an operating system

- (a) manages all the hardware resources of the computer
- (b) compiles a high-level program
- (c) Both (a) and (b)
- (d) None of the above

Ans. (b) compiles a high-level program

(viii) For a C program code for (i = 0; i ≤ 10; i++) {A}; A will run.

- (a) 10 times
- (b) 11 times
- (c) 12 times
- (d) None of the above

Ans. (b) 11 times

(ix) Which of the following is not a functional programming language

- (a) SML
- (b) HASKELL
- (c) C
- (d) LISP

Ans. (c) C

(x) A pointer in C language

- (a) is a address of some location
- (b) is useful in describing linked list
- (c) can be used to access the elements of an array
- (d) all of the above.

Ans. (d) all of the above.

2. State whether the following statements are True or False : (5 × 1 = 5)

- (i) Normal binary operators like + and - can be combined with assignment operator = to form new operators in C Language.

Ans. False

- (ii) A compiler translates a High-level program into a machine understandable language.

Ans. True

- (iii) An algorithm might never terminate.

Ans. False

- (iv) In C language pointers can be used as a function argument.

Ans. True

- (v) MS-WORD may be classified as an application software.

Ans. True

3. Fill in the blanks : (5 × 1 = 5)

- (i) ..... is used to open a file.

Ans. Topen

- (ii) ..... is used as a statement terminator in C.

Ans. Semicolom

- (iii) The operator && is an example for ..... operator.

Ans. Logical

- (iv) A function is called ..... when it calls itself.

Ans. Recursive

- (v) An Editor can be classified as ..... software.

Ans. Application

## SECTION - B

4. There are total six questions in this section. Attempt all questions. (6 × 5 = 30)

- (a) (i) Write a C program to swap two integer variables without using third variable.
- (ii) What is the difference between initialization and assignment of a variable?

Ans. (i) `#include < stdio.h >`

```
main()
{
    int x, y;
    printf("Please enter two numbers: \n"); scanf("%d%d", &x, &y);
    x = x + y;
    y = x - y;
    x = x - y;
    printf(" After swap value is :\n" );
    printf("x = %d y = %d", x , y );
    getch();
}
```

- (ii) The basic difference between initialization and assignment is that an assignment can be done as many times as desired whereas an initialization can be done only once.

*Some differences to point between initialization and assignment*

- a) you CANT assign to a const variable, whereas you can initialize it.

**For example :**

```
const AXC d = 3; //OK! initialization
d = 3; //WRONG! cannot assign to const
```

- b) initialization is about creating object assignment is about setting some value to object. This is why in the following code:

```
AXC d = 3;
AXC x;
x = 2;
```

**Another Example**

```
int i= 0; // definition and initialization of i
int j; // just a definition of j
j= i; // assignment of j
```

- (b) (i) **Differentiate between WHILE... DO and DO..WHILE loops.**

(ii) **Write a recursive C program to calculate the factorial of a given integer.**

Ans. (i) The difference between a "do ...while" loop and a "while { } " loop is that the while loop tests its condition before execution of the contents of the loop begins; the "do" loop tests its condition after it's been executed at least once. As noted above, if the test condition is false as the while loop is entered, the block of code is never executed. Since the condition is tested at the bottom of a do loop, its block of code is always executed at least once.

## **while**

The *while* loop is used to execute a block of code as long as some condition is true. If the condition is false from the start the block of code is not executed at all. The while loop tests the condition before at as executed so sometimes the loop may never be executed if initially the condition is not met. Its syntax is as follows.

```
while (tested condition is satisfied)
{
    block of code
}
```

In all constructs, curly braces should only be used if the construct is to execute more than one line of code. The above program executes only one line of code so it is not really necessary (same rules apply to if...else constructs) but you can use it to make the program seem more understandable or readable.

Here is a simple example of the use of the while loop. This program counts from 1 to 10.

```
#include <stdio.h>
int main(void)
{
    int count = 1;
    while (count <= 10)
    {
        printf("%d\n", count);
        count += 1;
    }
    return 0;
}
```

Note that no semi-colons (;) are to be used after the while (condition) statement. These loops are very useful because the condition is tested before execution begins. However it never seems to like these loops as they are not as clear to read as the do ...while loops. The while loop is the favourite amongst most programmers but as for me, I definitely prefer the do ...while loop.

## **do ....while**

The do...while loop also executes a block of code as long as a condition is satisfied.

Again, the difference between a “do ...while” loop and a “while {}” loop is that the while loop tests its condition before execution of the contents of the loop begins; the “do” loop tests its condition after it’s been executed at least once. As noted above, if the test condition is false as the while loop is entered the block of code is never executed. Since the condition is tested at the bottom of a do loop, its block of code is always executed at least once.

Some people don't like these loops because it is always executed at least once. When i ask them "so what?", they normally reply that the loop executes even if the data is incorrect. Basically because the loop is always executed, it will execute no matter what value or type of data is supposed to be required. The "do...while" loops syntax is as follows

```
do
{
    block of code
} while (condition is satisfied);
```

Note that a semi-colon (;) must be used at the end of the do ...while loop. This semi-colon is needed because it instructs whether the while (condition) statement is the beginning of a while loop or the end of a do ...while loop. Here is an example of the use of a do loop.

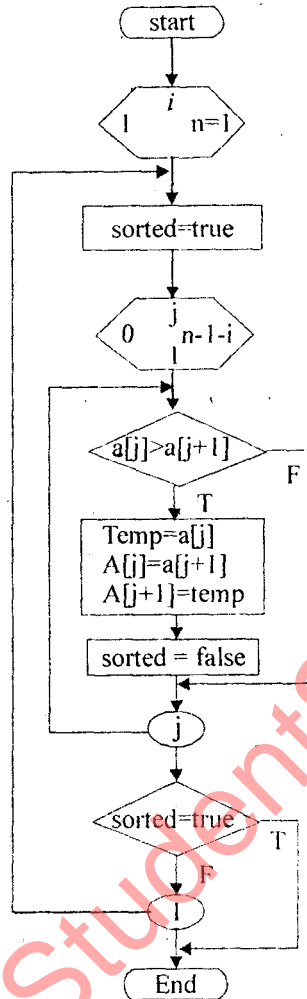
```
include <stdio.h>
int main(void)
{
    int value, r_digit; printf("Enter a number to be reversed.\n");
    scanf("%d", &value); do
    {
        r_digit = value % 10;
        printf("%d", r_digit);
        value = value / 10;
    } while (value != 0); printf("\n"); return 0;
}

(ii) #include <stdio.h>
long factorial(long);
main()
{
    long int number;
    printf("\nEnter an integer value: ");
    scanf("%ld", &number);
    printf("\nThe factorial of %ld is %ld\n", number, factorial(number));
}

long factorial(long x)
{
    if(x==1)
        return 1;
    else
        return x*factorial(x-1);
}
```

(c) (i) Draw a flow chart to sort three integers.

Ans.



(ii) What is dynamic memory allocation?  
Explain malloc function.

**Ans. Dynamic memory allocation :** The process of allocating memory at run time is known as dynamic memory allocation. Although c does not inherently have this facility there are four library routines which allow this function.

Many languages permit a programmer to specify an array size at run time. Such languages have the ability to calculate and assign during

executions, the memory space required by the variables in the program. But C inherently does not have this facility but supports with memory management functions, which can be used to allocate and free memory during the program execution. The following functions are used in C for purpose of memory management.

| Function | Task   |
|----------|--|
| malloc   | Allocates memory requests size of bytes and returns a pointer to the 1st byte of allocated space       |
| calloc   | Allocates space for an array of elements, initializes them to zero and returns a pointer to the memory |
| free     | Frees previously allocated space   |
| realloc  | Modifies the size of previously allocated space  |

**Memory allocations process :** According to the conceptual view the program instructions and global and static variable in a permanent storage area and local area variables are stored in stacks. The memory space that is located between these two regions is available for dynamic allocation during the execution of the program. The free memory region is called the heap. The size of heap keeps changing when program is executed due to creation and death of variables that are local for functions and blocks. Therefore it is possible to encounter memory overflow during dynamic allocation process. In such situations, the memory allocation functions mentioned above will return a null pointer.

**Allocating a block of memory :** A block of memory may be allocated using the function malloc. The malloc function reserves a block of memory of specified size and returns a pointer of type void. This means that we can assign it to any type of pointer. It takes the following form:

```
ptr=(cast-type*)malloc(byte size);
```

ptr is a pointer of type cast-type the malloc returns a pointer (of cast type) to an area of memory with size byte-size.

**Example :**

```
x=(int*)malloc(100*sizeof(int));
```

On successful execution of this statement a memory equivalent to 100 times the area of int bytes is reserved and the address of the first byte of memory allocated is assigned to the pointer x of type int.

- (d) (i) Write a C program to sequentially search a given integer element from a given list of numbers.
- (ii) What is the purpose of using structures in C? Explain with the help of a suitable example.

**Ans. (i)** #include<stdio.h>

```
#include<conio.h>
```

```
void linear_search(int[],int,int);
```

```
void main()
```

```
{
    int i,n,element[100],data,POS;
    clrscr( );
    printf("\n Please enter howmany elements do you  :");
    scanf("%d",&n);
    for (i=1;i<=n;i++)
    {
        printf("\n Enter a element :",i);
        scanf("%d",&element[i]);
    }
    printf("\n Enter a element to be searched: ");
    scanf("%d",&data) ;
    i=1, POS;
    POS = 0;
    while ((i<=n) || (POS==0))
    {
        if(data == element[i])
        {
            POS=i;
            printf("\n The data %d is found in the record at position %d.
            ", data,POS); i=i+1;
            continue;
        }
    }
}
```

```

else
    i = i + 1;
    if(i==n)
        break;
}
if(POS == 0)
printf("\nThe data %d is not found in the record. :",data);
getch();
}

```

(ii) The purpose of structure to represents the real life data with the help of primitive data types. Since all the real life data is not able to store by primitive data types. So, you can use the concept of structure to declare real time data. For example, the following some real life data is not possible to declare through primitive data types.

**Date :** A date is combination of day, month and year

**Address :** An address of a person consists of name, house number, city, state & pin code.

**Account details :** An account contains the account number, name of bank, balance amount.

**Book :** A book contains name of book, name of author, name of publisher & price.

In all above example, it is not possible to declare by primitive data types. So you have to create new data type to store all these real life data.

A structure is a set of interrelated data. A structure in C is a set of primitive data types, which are grouped together to form a new data types. A structure is mechanism provided by the language to create complex data types.

**Declaring and using a structure :** In programming language C, a structure can be declared using the *struct* keyword. The set of variables that form the structure must be declared with a valid name similar to declaring variables. Each variable inside a structure can be of different data type.

Let us consider a structure, which can be used to represent date. Date is a simple data structure but not available in C. A date has three components.

day of month (integer, range 1-31)

month (integer, range 1-12)

year (integer, four digit)

Based on the above information, we can see that all the three components of date can be of integer type. The date structure can be created as follows :

```
struct date
```

```
{
    int day;
    int month;
    int year;
};
```



Each variable declare inside a structure is known as a member variable. In the date *day, month, year* are member variables.

Each member variable in structure can be accessed individually. Once a structure is declared, it can be used as a primitive data types.

For example :

```
struct date todays_date;
```

defines a variable called todays\_date to be of the same data type as that of the newly defined data type struct date.

**Assigning values to structure elements :** To assign todays date to the individual elements of the structure todays\_date, the statement

```
todays_date.day = 21;
todays_date.month = 07;
todays_date.year = 1985;
```

is used.

```
#include <stdio.h>
struct date {
    int day;
    int month;
    int yea
};
main()
{
    struct date today;
    today.day = 10;
    today.month = 4;
    today.year = 1995;
    printf("Todays date is %d/%d/%d.\n", \
    today.day, today.month, today.year);
}
```

(e) (i) Find the value of X in the equation  $(1230)_4 = X_6$ .

(ii) Draw the functional block diagram of a Digital Computer and discuss its components in brief.

**Ans. (i)**

**Ans. (ii)** A digital computer is used to store data in terms of digits and proceeds in discrete steps from one state to the next. The states of a digital computer typically involve binary digits which may take the form of the presence or absence of magnetic markers in storage medium, on-off switches. In digital computers, even letters, words and whole texts are represented digitally. A digital

computer can only approximate a continuum by assigning large numbers of digits to a state description and by proceeding in arbitrarily small steps. A digital computer consists of the following main components:

- Input devices
- Memory
- Central Processing Unit (CPU)
- Output devices

The brain of computer is Central Processing Unit (CPU).

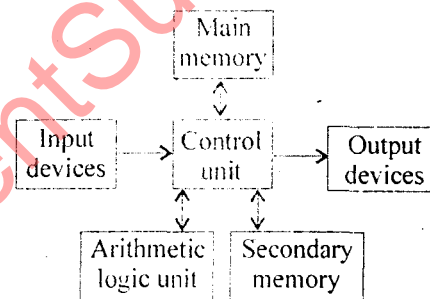
A CPU has mainly three components:

- Main-memory
- Arithmetic Logic Unit (ALU)
- Control Unit

The main memory is used to hold the data being processed and holds the instruction for doing the processing.

ALU is used to perform only arithmetic and logic operations. The arithmetic operations are addition, subtraction, multiplication and division and logic operation are OR, AND and NOT.

Control unit is used to monitor and control all the operations of the computer. It also acts as a interface between main memory and ALU, and main memory and secondary memory.



**Fig.** Block diagram of a digital computer

Thus, the original programming language fed into the computer must be first converted by means of an interpreter or a compiler into a machine-readable assembly language. Only then the machine is able to execute the fed-in instructions.

The input is coded up in memory, which is a grid of electronic on-off switches. The processor, which is a chip of integrated circuitry, alters what is in the memory, resulting in a different on-off pattern of switches, and then the output decodes and displays the new contents of the memory. So the actual computation consists of the processor's activities on the memory. Accordingly the processor and the memory stand in mutual contact with each other.

### SECTION - C

5. This section contains SEVEN programming question. Attempt any FIVE questions. All answers must contain Flow chart/Algorithm for your program logic: (10×5=50)

(a) Write a C program to read in 10 integer numbers and print their average, minimum and maximum numbers.

```
Ans. #include <stdio.h>
main( )
{
    int num[10], i, sum=0,min ,max;
    float avg;
    printf(" Enter The value of numbers\n");
    for(i=0; i<10; i++)
        scanf("%d",&num[i]);
    sum = sum + num[i];
    avg = (float) sum/10;
    min=num[0];
    for(i=1;i<10;i++)
        if(min > num[i])
            min = num[i];
    max=num[0];
    for(i=1;i<10;i++)
        if(max < num[i])
            max= num[i];
    printf("The average = %f\n", avg);
    printf("The minimum = %d\n",min);
    printf("The maximum = %d\n", max);
}
```

(b) Write a C program to add, multiply two N × N matrix.

```
Ans. #include <stdio.h>
main()
{
    int a[][] , b[][]
    static int add[][] , mul[][];
    int r, c, N, sum
    printf("PI. enter the number");
```

```

scanf("%d",&N );
for(r=0; r<N; r++)
for(c=0; c<N; c++)
    scanf("%d",&a[r][c] );
for(r=0; r<N; r++)
for(c=0; c<N; c++)
    scanf("%d",&b[r][c] );
for(r=0; r<N; r++)
for(c=0; c<N; c++)
    add[r][c]=a[r][c]+b[r][c];
printf("The addition of Two matrix is : ");
for(r=0; r<N; r++)
{
    for(c=0; c<N; c++)
        printf(" %5d",add[r][c]);
    printf("\n");
}

for(r=0; r<N; r++)
for(c=0; c<N; c++)
for(k=0; k<N; k++)
    mul[r][c]=mul[r][c]+a[r][k]*b[k][c];
printf("The multiplication of Two matrix is:");
for(r=0; r<N; r++)
{
    for(c=0; c<N; c++)
        printf("%5d",mul[r][c]);
    printf("\n");
}
getch();
}

```

- (c) Write a simple database program in C which stores personal details of 100 persons such as Name, Date of Birth, Address, Phone number etc.

**Ans.** #include<stdio.h>  
#include<conio.h>  
#include<string.h>

```

    struct date
    {
        int dd;
        int mm;
        int yyyy;
    };

    struct person_detail
    {
        char name[15];
        struct date DOB;
        char add[25];
        long int phone;
    };

    main()
    {
        int i;
        struct person_detail ps[100];
        clrscr ();
        printf("Please enter the 100 persons detail\n");
        for(i=0;i<2;i++)
        {
            fflush(stdin);
            printf(" \nEnter %d name: ",i+1);
            gets(ps[i].name);
            printf(" \nEnter %d DOB: ",i+1);
            scanf("%d%d%d",&ps[i].DOB.dd,&ps[i].DOB.mm,&ps[i].DOB.yyyy);
            printf(" \nEnter %d add: ",i+1);
            fflush(stdin);
            gets(ps[i].add);
            printf(" \nEnter %d Phone: ",i+1);
            scanf("%ld", &ps[i].phone);
        }

        for(i=0;i<2;i++ )
        {
            printf("\n");

```

```

        puts(ps[i].name);
        printf("%d : %d: %d\n ",ps[i]. DOB. dd,ps[i]. DOP. mm,ps[i].DOB.
        yyyy);
        puts(ps[i].add);
        printf("%ld",ps[i]. phone);
        printf("\n");
    }

    getch() ;
}

```

- (d) Write a C program which reverses the digits of the integer input given to it. For example an input 65367 is outputted as 76356.

Ans. #include<stdio.h>  
#include<math.h>  
#include<conio.h>  
main()  
{  
int num,reverse,rem;  
scanf("%d",&num);  
printf("%d", num);  
while(num>0)  
{  
rem = num%10;  
num = num/10;  
reverse = reverse\*10 + rem;  
}  
printf("The reverse number = %d", reverse);  
getch();  
}

- (e) Write a C program to calculate the sum of the following series upto 50 terms

$$\text{SUM} = -1^3 + 3^3 - 5^3 + 7^3 - 9^3 + 11^3 - \dots$$

Ans. #include<stdio.h>  
#include<math.h>  
#include<conio.h>  
main()  
{  
int term,i,j;  
long int sum=0;

```

    clrscr();
    i=1,j=1;
    while(i<100)
    {
        sum=sum +pow(-1,j)*pow(i,3);
        j=j+1;
        i=i+2;
    }
    printf("sum= %ld",sum);
    getch();
}

```

**(f) Write a C program to print  $n^{\text{th}}$  Fibonacci number.**

**Ans.** #include<stdio.h>

#include<math.h>

#include<conio.h>

main()

```

{
    int a,b,i,n;
    long int term;
    clrscr();
    printf("Enter the nth position:");
    scanf("%d",&n);
    a=0;b=1;
    i=1;
    while(i<n-1)
    {
        term=a+b;
        a=b;
        b=term;
        i=i+1;
    }
    printf(" term= %ld",term);
    getch();
}

```

**(g) Write a C program to arrange given  $n$  strings in lexicographical order.**

**Ans.** #include<stdio.h>

#include<conio.h>

#include<string.h>

```

void main()
{
    int i,j,d;
    char ch[20],temp[30];
    clrscr();
    printf("\nEnter the no.of string:");
    scanf("%d",&d);
    printf("\nEnter the strings:");
    for(i=0;i<d;i++)
    {
        gets(ch[i])
    }
    printf("\nThe strings are:");
    for(i=0;i<d;i++)
    {
        printf("\n%s",ch[i]);
    }
    for(i=0;i<d;i++)
    {
        for(j=i+1;j<d;j++)
        if(strcmp(ch[i],ch[j])>0)
        {
            strcpy(temp,ch[i]);
            strcpy(ch[i],ch[j]);
            strcpy(ch[i],temp);
        }
    }
    getch();
}

```