STUCTURED PROGRAMMING APPROACH (DEC 19)

Q.P.CODE: 76907

Q.1.a) Attempt the Multiple choice Questions. (6 M)

i. The format identifier '%i' is also used for ______ data type?

a) Char b) int c) double d) float

Ans: b) int

ii. Which keyword can be used for coming out of recursion?

a) Break b) exit c) return d) all of above

Ans: b) exit

iii. What will happen if in a C program you assign a value to an array element whose subscript exceeds the size of array?

a) The element will be set to 0
b) The compiler would report an error
c) The program may crash if some important data gets overwritten
d) The array size would appropriately grow.

Ans: d) the array size would appropriately grow

iv. A pointer is

a) A keyword used to create variable
b) A variable that stores address of an instruction
c) A variable that stores address of other variable
d) All of the above

Ans: c) A variable that stores address of other variable.

v. In which order do the following gets evaluated

1. Relational
2. Arithmetic
3. Logical
4. Assignment

a) 2134 b) 1234 c) 4321 d) 3214

Ans: a) 2134

vi. Which of the following cannot a be structure member?

a) Another structure b) Function c) Array d) None of the mentioned

Ans: a) Another structure.

---

b) Find the output of the following: **(4 M)**

i.
```
#include<stdio.h>
Void main()
{
 int c;
 for(c=1;c<=5;)
 printf("%d",++c);
}
```
Ans:

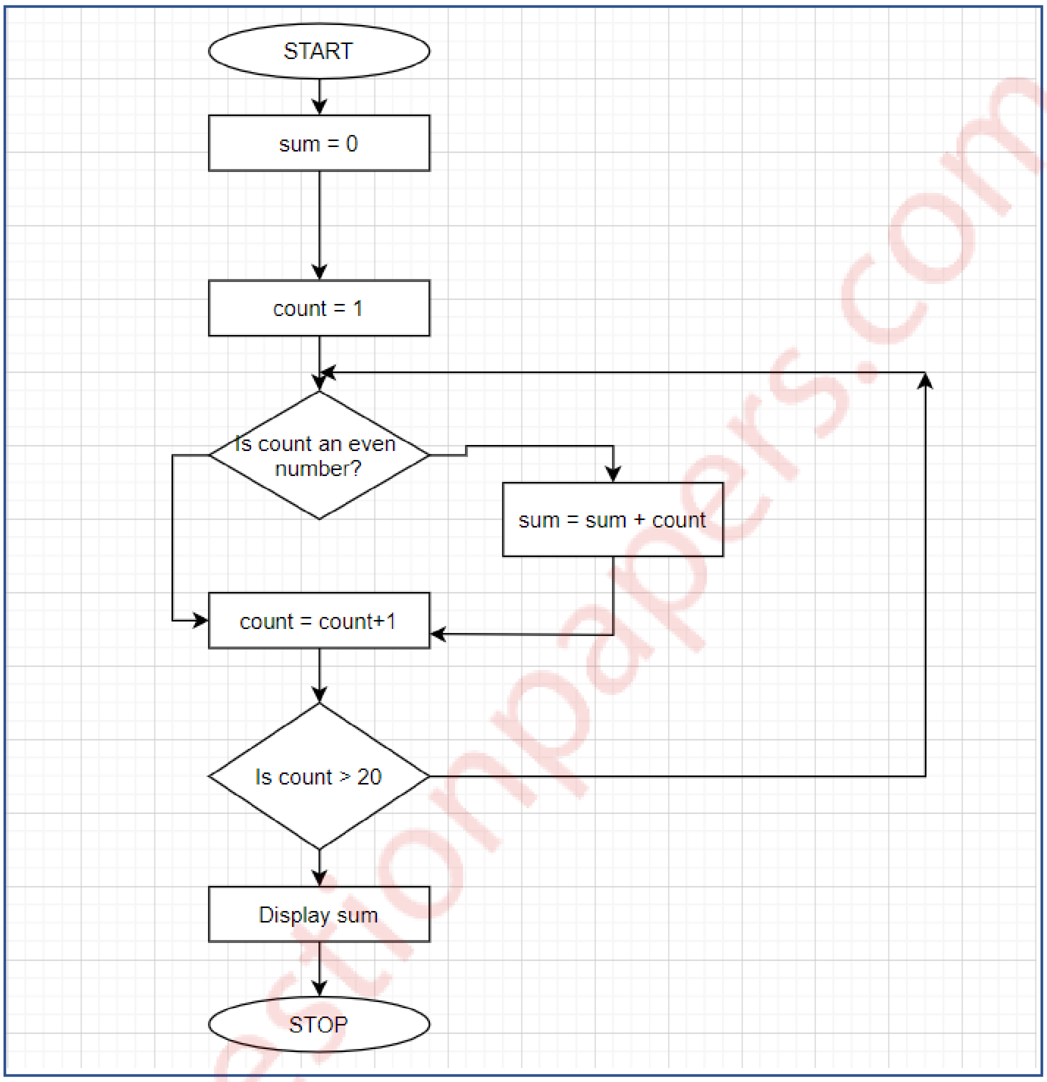
```
2 3 4 5 6
```

ii. **How many 'x' are printed by the following code?**
```
#include<stdio.h>
Void main()
{
 Int i=5;
 While(i-->0)
 Printf("x");
 Printf("x");
}
```
Ans: 6

---

c) Draw a flowchart for printing the sum of even terms contained within the numbers 0-20. (4 M)

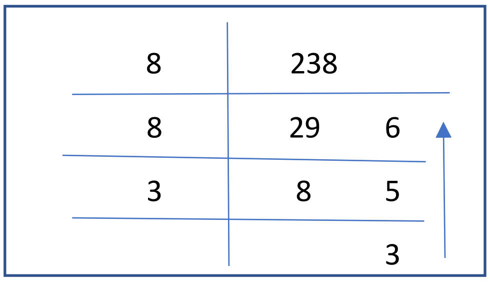
Ans:



d) Solve the following (6 m)

i. Convert 238 decimal to octal

Ans:



8	238	
8	29	6
3	8	5
		3

Then, when we put the remainders together in reverse order, we get the answer. The decimal number 238 converted to octal is therefore: 356.

ii. Convert A3D Hexadecimal to decimal

Ans:

$(A3D)_{16} = (10 \times 16^2) + (3 \times 16^1) + (13 \times 16^0) = (2621)_{10}$

Q.2)

a) Distinguish Between (6 M)

a) While and do-while loop

Ans:

While Loop	**Do-While Loop**
Condition is checked first then statement(s) is executed.	Statement(s) is executed atleast once, thereafter condition is checked.
It might occur statement(s) is executed zero times, If condition is false.	At least once the statement(s) is executed.
No semicolon at the end of while. while(condition)	Semicolon at the end of while. while(condition);
If there is a single statement, brackets are not required.	Brackets are always required.
Variable in condition is initialized before the execution of loop.	variable may be initialized before or within the loop.
while loop is entry controlled loop.	do-while loop is exit controlled loop.
while(condition) { statement(s); }	do { statement(s); } while(condition);

---

ii) break and continue

Ans:

Break	**Continue**
A break can appear in both switch and loop (for, while, do) statements.	A continue can appear only in loop (for, while, do) statements.
A break causes the switch or loop statements to terminate the moment it is executed. Loop or switch ends abruptly when break is encountered.	A continue doesn't terminate the loop, it causes the loop to go to the next iteration. All iterations of the loop are executed even if continue is encountered. The continue statement is used to skip statements in the loop that appear after the continue.
The break statement can be used in both switch and loop statements.	The continue statement can appear only in loops. You will get an error if this appears in switch statement.

When a break statement is encountered, it terminates the block and gets the control out of the switch or loop.	When a continue statement is encountered, it gets the control to the next iteration of the loop.
A break causes the innermost enclosing loop or switch to be exited immediately.	A continue inside a loop nested within a switch causes the next loop iteration.

b) Write a C program that will convert a decimal number into any base.

(6 M)

Ans:

```
#include <math.h>
#include <stdio.h>
long long convert(int n);
int main() {
  int n;
  printf("Enter a decimal number: ");
  scanf("%d", &n);
  printf("%d in decimal = %lld in binary", n, convert(n));
  return 0;
}
long long convert(int n) {
  long long bin = 0;
  int rem, i = 1, step = 1;
  while (n != 0) {
    rem = n % 2;
    printf("Step %d: %d/2, Remainder = %d, Quotient = %d\n", step++, n, rem, n /
2);
    n /= 2;
    bin += rem * i;
    i *= 10;
  }
  return bin;  }
```

Output:

```
Enter a decimal number: 19
Step 1: 19/2, Remainder = 1, Quotient = 9
Step 2: 9/2, Remainder = 1, Quotient = 4
Step 3: 4/2, Remainder = 0, Quotient = 2
Step 4: 2/2, Remainder = 0, Quotient = 1
Step 5: 1/2, Remainder = 1, Quotient = 0
19 in decimal = 10011 in binary
```

c) Write a C program to calculate the sum of following series without pow() library function S=1-x+x*2/2!-x*3/3!+…………N terms. (8 M)

Ans:

```
#include <stdio.h>
void main()
{
    float x,sum,no_row;
    int i,n;
    printf("Input the value of x :");
    scanf("%f",&x);
    printf("Input number of terms : ");
    scanf("%d",&n);
    sum =1; no_row = 1;
    for (i=1;i<n;i++)
    {
      no_row = no_row*x/(float)i;
      sum =sum+ no_row;
    }
    printf("\nThe sum  is : %f\n",sum);
}
```

Output:

```
Input the value of x :3
Input number of terms : 5
The sum  is : 16.375000
```

Q.3)

a) What is an array? What does an array name signify? Can array index be negative? Write a C program to arrange the number stored in an array in such a way that array will have the odd numbers followed by even numbers. (10 M)

Ans:

- An array is a data structure that contains a group of elements. Typically these elements are all of the same data type, such as an integer or string. Arrays are commonly used in computer programs to organize data so that a related set of values can be easily sorted or searched.
- Yes, array index can be negative. It can be used to print the array in a reverse order.

  Program:

```
#include <conio.h>

 int main()
{
    int a[10000],b[10000],i,n,j,k,temp,c=0;

    printf("Enter size of the  array : ");
    scanf("%d", &n);
    printf("\nEnter elements in array : ");
    for(i=0; i<n; i++)
    {
        scanf("%d",&a[i]);
        if(a[i]%2==1)
         c++;
    }
    for(i=0; i<n-1; i++)
    {

        for(j=0; j<n-i-1; j++)
        {
          if(a[j]>a[j+1])
          {
                temp=a[j];
                a[j]=a[j+1];
                a[j+1]=temp;
                   }

        }
    }

      k=0;
    j=n-c;

      for(i=0; i<n; i++)
    {
       if(a[i]%2==0)
       {
                if(k<n-c)
                  b[k++]=a[i];
                }
                else
                {
                        if(j<n)
                  b[j++]=a[i];
         }
    }

    printf("\narray after sorting even and odd elements separately:\n ");

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

Output:

```
Enter size of the  array : 9
Enter elements in array : 1 3 5 7 9 2 4 6 8
array after sorting even and odd elements separately:
 1 2 3 4 5 6 7 8
```

b) Write a program that accepts a word from the user and prints in the following way. For ex. If the word is "STUDY" the program will print it as

S
ST
STU
STUD
STUDY **(10 M)**

Ans:

```
#include <stdio.h>

#include <string.h>

int main()

{

  char* a=”STUDY”;

  int i,j;

  for(i=0; i<strlen(a); i++)

  {

   for(j=0; j<=i; j++)

   {

    printf("%c",a[j]);

   }

   printf(" ");

}
```

Output:

```
S
ST
STU
STUD
STUDY
```

Q.4)

a) **What is string? Explain the use of gets()? Write a c program that will read a word and rewrite it in alphabetical order. For ex. If the word is "matrix" the program should print "aimrtx". (10 M)**

Ans:

- Strings are defined as an array of characters. The difference between a character array and a string is the string is terminated with a special character '\0'.
- **gets():**
- Gets function is used to scan a line of text from a standard input device.
- This function will be terminated by a newline character. The nwline character won't be included as part of the string. The string may include white space characters.
- **Syntax :**

  char *gets(char *s);
- This function is declared in the header file stdio.h. It takes a single argument.
- The argument must be a data item representing a string. On successful completion, shall return a pointer to string s.

```
#include<stdio.h>
 #include<conio.h>
 int main()
 {
      char str[100],temp;
      int i,j;
      clrscr();
      printf("Enter the string :");
      gets(str);
      printf("%s in ascending order is -> ",str);
      for(i=0;str[i];i++)
      {
           for(j=i+1;str[j];j++)
           {
                if(str[j]<str[i])
                {
                     temp=str[j];
                     str[j]=str[i];
                     str[i]=temp;
                }
           }
      }
      printf("%s\n",str);
      getch();
```

```
    return 0;
}
```

Output:
```
Enter the String: matrix
matrix in an ascending order is -> aimrtx
```

---

b) Explain recursion and its advantages? Write a recursive c program to find the factorial of a given number. (10 M)

Ans:

- Recursion: A function that calls itself is called as recursive function and this technique is called as recursion.
- Advantages:
    a. Reduce unnecessary calling of functions.
    b. Through Recursion one can solve problems in easy way while its iterative solution is very big and complex.
    c. Extremely useful when applying the same solution.

Program:
```
#include<stdio.h>
long int multiplyNumbers(int n);
int main() {
  int n;
  printf("Enter a positive integer: ");
  scanf("%d",&n);
  printf("Factorial of %d = %ld", n, multiplyNumbers(n));
  return 0;
}
long int multiplyNumbers(int n) {
  if (n>=1)
    return n*multiplyNumbers(n-1);
  else
    return 1;
}
```

Output:
```
Enter a positive integer: 6
Factorial of 6 = 720
```

Q.5)

a) Explain the storage classes with example. (10 M)

Ans:

- The different locations in the computer where we can store data and their accessibility, initial values etc. very based on the way they are declared. These different ways are termed as different storage classes.
- In C there are for storage classes, namely
  1. Automatic
  2. Register
  3. Static
  4. External or global
- Let us see these storage classes one by one

1. **Automatic storage class**
   In this case data is stored in memory
   The initial value of such a variable is garbage
   The scope of the variable is local i.e. limited to the function in which it is defined.
   The life of such variables is till the control remains in the particular function where it is defined.
   For e.g.:
   Int i; or auto int i;

2. **Register storage class**
   In this case data is stored in CPU register
   The initial value of such a variable is garbage.
   The scope of the variable is local i.e. limited to the function in which it is defined
   The life of such variables is till the control remains in the particular function where it is defined.
   For e.g.:
   Register int I;
   In this case the data is stored in a small memory inside the processor called its registers.
   The advantage of such storage class is that since the data is in the processor itself, its access and operation on such data is faster.
   There is limitation on the size of the data that can declared to be register storage class. The data should be such that it doesn't require more than 4 bytes. Hence double and long double data types cannot be declared as a register.
   Also there is a limitation on the maximum number of variables in a function that can be a register class. The limitation is that a maximum of 3 register class variable can be declared in a function.

3. **Static storage class**
   In this case data is stored in a memory

The initial value of such a variable is zero
The scope of the variable is local i.e. limited to the function in which it is defined
The life of such variable is till the program is alive.
For e.g.:
Static int I;
If a variable is declared static, its value remains unchanged even If the function execution is completed.
When the execution to that function returns, the previous value is retained.
Thus it says the initialization is only once. If you have an initialization statement of a static member, it will be executed only once i.e. for the first time when this function is called.

4. **External or global storage class**
   In this case data is stored in memory
   The initial value of such a variable is zero.
   The scope of the variable is global i.e. it is accessible from anywhere in the program.
   The life such a variable is till the program is alive.

---

b) Declare a structure to store the information of 10 cricketers.

- **i. Cricketer name**
- **ii. Matches Played**
- **iii. Runs Scored**
- **iv. Strike Rate**

Use a function to display the cricketer information having the maximum strike rate. (10 M)

Ans:

Program:

```
#include<stdio.h>
#include<conio.h>
struct Cricketer
{
 name[50];
 int matches;
 int runs;
 float strike;
void main()
{
 struct Cricketer e[100];
 int n,i,maximum = 0;
 clrscr();
 printf("Enter the number of Cricketers");
 scanf("%d",&n);
 for(i=0;i<=n-1;i++)
 {
  printf("\nEnter name,matches,runs, strike");
  scanf("%s%d%d%d",&e[i].name,&e[i].matches,&e[i].runs,&e[i].strike);
 }
  printf("\nName\t\tMatches\t\tRuns\t\tStrike\n");
  printf("--------------------------------------------------------------------------------");
  for(i=0;i<=n-1;i++)
  {
   printf("%s%d%d%d",&e[i].name,&e[i].matches,&e[i].runs,&e[i].strike);
  }
Maximum = e[i].strike;
For(i=0;i<=n-1;i++)
{
if(e[maximum].strike < e[i].strike)
      highest = i;
}
printf("/nThe maximum strike rate is %d",e[maximum].strike);
  getch();
}
```

Output:

```
Enter number of Cricketers 10
Enter Name, matches, runs, strike
John
100
20000
35.5

Enter Name, matches, runs, strike
sam
101
20200
37.5

Enter Name, matches, runs, strike
Miller
105
20080
38.5
```

```
Enter Name, matches, runs, strike
Dale
107
20400
36.5

Enter Name, matches, runs, strike
Smith
105
20700
39.5

Enter Name, matches, runs, strike
david
106
28000
40.5

Enter Name, matches, runs, strike
Dhoni
110
20500
42.5

Enter Name, matches, runs, strike
Johny
107
20600
35.5

Enter Name, matches, runs, strike
kohli
120
20000
36.5

Enter Name, matches, runs, strike
Rohit
103
20700
36.5
```

```
Name          Matches      Runs     Strike
------------------------------------------
John          100          20000    35.5
Sam           101          20200    37.5
Miller        105          20080    38.5
Dale          107          20400    36.5
Smith         105          20700    39.5
David         106          28000    40.5
Dhoni         110          20500    42.5
Johny         107          20600    35.5
kohli         120          20000    36.5
Rohit         103          20700    36.5
The maximum strike rate is 42.5
```

Q.6)

a) How do pointers differ from variable in C? Write a c program to add two pointers. (10 M)

Ans:

- Pointers are variables that are used to store the address of another variable.
- Address of a variable is the memory location number which is allotted to the variable.
  The memory addresses are 0, 1, 2, 3… and so on up to the capacity of the memory. The address is normally displayed in hexadecimal form. Hexadecimal form is a representation of number somewhat similar to binary number. Here four binary digits are combined together to form a hexadecimal number.
- Pointers unlike other variables do not store values. As stated they store the address of other variables.
- It is already mentioned in the first statement that pointers are also variables. Hence, we can also have a pointer that is pointing to another pointer.
- Syntax of pointer declaration : Data_type *ptr_name;
- Wherein "Data_type" is the data type of the variable to which the pointer is supposed to point. If we want a pointer to point to an integer than, we need to have the data type of the pointer as "int", for a float type data pointer should also be of the "float" type and so on.
- The "ptr_name" is an identifier i.e. the name of the pointer. The same rules of identifiers apply to the pointer name as to any other variable declaration. The most important difference in the declaration of a pointer is the "*" sign given before the pointer name.
- Hence, according to the syntax seen above, if we want to declare a pointer for "int" type data then we can declare it as given in the example below: int *p; Here,

the pointer name is “p”. Hence, “p” can be used as a pointer to point to any of the variable of type “int”.

- Syntax : data_type *var_name;
- Example : int *p; char *p;

Program:

```
#include <stdio.h>
int main()
{
  int first, second, *p, *q, sum;

printf("Enter two integers to add\n");
scanf("%d%d", &first, &second);

p = &first;
q = &second;

sum = *p + *q;

printf("Sum of the numbers = %d\n", sum);

return 0;
}
```

Output:

```
Enter two integers to add
4
5
Sum of entered number is 9
```

---

b) What is file? Write a c program that include the menu that must have the following capabilities

i. Enter the several lines of text and store them in data file.

ii. Retrieve and display the particular line

iii. Delete n lines **(10 M)**

Ans:

- For file handling or accessing the contents of file, there are certain predefined functions available in the C programming language.

- An important thing required to access files is the "FILE pointer". This pointer is used to point to the values stored in the file. A file pointer is hence to be created for accessing the files. The syntax for creating a file pointer is as given below: FILE *<identifier for pointer>; For e.g. FILE *fp;
- Hence in every program we write in this section to access files, we will use this kind of pointer declaration. This pointer is used to point the data to be accessed in the file i.e. whenever a data is read or written in the file, it is from the location pointed by the file pointer "fp".

Program:

```
#include <stdio.h>
int main ()
{
 FILE * fptr;
 int i,n;
 char str[100];
 char fname[20]="test.txt";
 char str1;
           printf(" Input the number of lines to be written : ");
           scanf("%d", &n);
           printf("\n :: The lines are ::\n");
           fptr = fopen (fname,"w");
           for(i = 0; i < n+1;i++)
              {
              fgets(str, sizeof str, stdin);
              fputs(str, fptr);
              }
 fclose (fptr);
/*-------------- read the file -------------------------------------*/
           fptr = fopen (fname, "r");
           printf("\n The content of the file %s is  :\n",fname);
           str1 = fgetc(fptr);
           while (str1 != EOF)
              {
                    printf ("%c", str1);
                    str1 = fgetc(fptr);
              }
  printf("\n\n");
  fclose (fptr);
/*-----------------------------------------------------------------------------*/
    FILE *fp1, *fp2;
     char filename[40];
     char c;
     int del_line, temp = 1;
printf("Enter file name: ");
scanf("%s", filename);
fp1 = fopen(filename, "r");
c = getc(fp1);
while (c != EOF)
{
  printf("%c", c);
  c = getc(fp1);
}
//rewind
rewind(fp1);
printf(" \n Enter line number of the line to be deleted:");
scanf("%d", &del_line);
fp2 = fopen("copy.c", "w");
c = getc(fp1);
while (c != EOF) {
  c = getc(fp1);
  if (c == '\n')
  temp++;
  if (temp != del_line)
  {

    putc(c, fp2);
  }
}
fclose(fp1);
fclose(fp2);

remove(filename);
rename("copy.c", filename);
printf("\n The contents of file after being modified are as  follows:\n");
fp1 = fopen(filename, "r");
c = getc(fp1);
while (c != EOF) {
    printf("%c", c);
    c = getc(fp1);
}
fclose(fp1);
return 0;
}
```

Output:

```
Input the number of lines to be written : 4
 :: The lines are ::
test line 1
test line 2
test line 3
test line 4
 The content of the file test.txt is  :
test line 1
test line 2
test line 3
test line 4
Enter line number of the line to be deleted: 2
The content of the file after being modified are as follows:
test line 3
test line 4
```