ಕುವೆಂಪು ವಿಶ್ವವಿದ್ಯಾಲಯ KUVEMPU UNIVERSITY

ಪರೀಕ್ಷಾ ಕೊಠಡಿಯಲ್ಲಿ ಅಭ್ಯರ್ಥಿಗಳು ಅನುಸರಿಸಬೇಕಾದ ಸೂಚನೆಗಳು
INSTRUCTIONS TO CANDIDATES (at the examination hall)

1. ಪ್ರವೇಶ ಪತ್ರದ ಹಿಂಭಾಗದಲ್ಲಿ ಮುದ್ರಿಸಿರುವ ಸೂಚನೆಗಳನ್ನು ತಪ್ಪದೇ ಅನುಸರಿಸಬೇಕು.
   Please follow the instructions given in the Admission Ticket Carefully.
2. ಉತ್ತರಿಸುವ ಮುನ್ನ ಉತ್ತರ ಪತ್ರಿಕೆಯಲ್ಲಿ 40 ಪುಟಗಳು ಇದೆಯೇ ಎಂಬುದನ್ನು ಖಚಿತ ಪಡಿಸಿಕೊಳ್ಳಿ, ಸರಿಯಾಗಿ ಇಲ್ಲದಿದ್ದಲ್ಲಿ, ಕೊಠಡಿಯ ಮೇಲ್ವಿಚಾರಕರಿಂದ ಮತ್ತೊಂದು ಉತ್ತರ ಪತ್ರಿಕೆಯನ್ನು ಪಡೆಯಬೇಕು.

   Before answering the questions, ensure that the answer book contains 40 pages. In case it is defective, request the invigilator to issue a new answer book.
3. ಪ್ರವೇಶ ಪತ್ರಿಕೆ ಮತ್ತು ಪ್ರಶ್ನೆ ಪತ್ರಿಕೆಯಲ್ಲಿ ನಮೂದಿಸಿರುವ ಪ್ರಶ್ನೆ ಪತ್ರಿಕೆಯ ಕೋಡ್ ಸರಿಯಿದೆಯೆ ಎಂಬುದನ್ನು ಖಚಿತಪಡಿಸಿಕೊಳ್ಳಿ.

   Ensure that the QP code on Admission Ticket and actual Question Paper are identical.
4. ಅಭ್ಯರ್ಥಿಗಳು ಪರೀಕ್ಷೆಗೆ ಸಂಬಂದ ಪಟ್ಟ ವಿಷಯಗಳಾದ ಡಿಗ್ರಿ, ವಿಷಯ, ಹೆಸರು, ರಿಜಿಸ್ಟರ್ ನಂಬರ್, ಪರೀಕ್ಷಾ ದಿನಾಂಕ, ಸ್ಟುಡೆಂಟ್ ಐ.ಡಿ. ಹಾಗೂ ಪ್ರಶ್ನೆ ಪತ್ರಿಕೆಯ ಕೋಡ್‌ನ್ನು (ಕ್ಯೂ ಪಿ ಕೋಡ್) 1ನೆಯ ಪುಟದಲ್ಲಿರುವ ಭಾಗ-Iರಲ್ಲಿ ಬರೆಯಬೇಕು

   Candidates are instructed to write information such as Degree, Exam, Subject, Candidate name, **Register No.**, Exam date, **Student ID and Question paper code (QP Code)** in Part-I.
5. ಸ್ಟುಡೆಂಟ್ ಐ.ಡಿ. ಹಾಗೂ ಪ್ರಶ್ನೆ ಪತ್ರಿಕೆಯ ಕೋಡ್‌ನ್ನು (ಕ್ಯೂ ಪಿ ಕೋಡ್) ನಿಗದಿ ಪಡಿಸಿದ ಬಾಕ್ಸ್‌ನಲ್ಲಿ ಬರೆಯಬೇಕು ಮತ್ತು ಕೆಳಗೆ ಇರುವ ಸಂಬಂಧಪಟ್ಟ ಸರ್ಕಲ್‌ಗಳನ್ನು ಶೇಡ್ ಮಾಡಬೇಕು. ಇದರ ನಮೂನೆಯನ್ನು 2ನೆಯ ಪುಟದಲಿ ತೋರಿಸಲಾಗಿದೆ.

   Student ID and QP code are to be written in the designated boxes. Appropriate ovals below these boxes are to be shaded as shown in the sample in Page 2
6. ನಕ್ಷೆ ಅಥವಾ ಗ್ರಾಫ್ ಹಾಳೆಗಳನ್ನು ಉಪಯೋಗಿಸಿದಲ್ಲಿ ಅವುಗಳ ಮೇಲೆ ರಿಜಿಸ್ಟರ್ ನಂಬರ್ ಹಾಗೂ ಸ್ಟುಡೆಂಟ್ ಐ.ಡಿ. ಬರೆಯಬಾರದು.

   Do not write Register No. / Student ID on map / graph sheet, if used.
7. ಉತ್ತರ ಪತ್ರಿಕೆಯ ಕ್ರಮಸಂಖ್ಯೆಯನ್ನು ಕೊಠಡಿಯ ಮೇಲ್ವಿಚಾರಕರ ದಿನಚರಿಯಲ್ಲಿ ಬರೆದು ಪೂರ್ಣ ಸಹಿ ಮಾಡಬೇಕು.

   Candidates shall write the serial number of the Answer books and affix their signature, in token of their receiving the answer book in the invigilator's Dairy.
8. ಉತ್ತರಗಳನ್ನು ನೀಲಿ ಅಥವಾ ಕಪ್ಪುಶಾಹಿಯ ಪೆನ್ನನ್ನು ಘಾತ್ರ ಉಪಯೋಗಿಸಿ ಬರೆಯಬೇಕು. ಉತ್ತರ ಪತ್ರಿಕೆಗಳ ಜೆರಾಕ್ಸ್ ಪ್ರತಿ ಇಚ್ಚಿಸುವವರು ಉತ್ತರಗಳನ್ನು ಕಪ್ಪುಶಾಹಿಯಲ್ಲಿ ಮಾತ್ರ ಬರೆಯಬೇಕು.

   Answers must be written in black or blue ink. Those who desire to obtain xerox copy of their answer books must write in black ink.
9. ಉತ್ತರ ಪತ್ರಿಕೆಯಲ್ಲಿಖಾಲಿ ಉಳಿಯುವ ಎಲ್ಲಾಹಾಳೆಗಳ ಮೇಲೆ ಉದ್ದನೆಯ ಗೆರೆ ಎಳೆಯಬೇಕು.
   Score off all the blank pages left in the answer books at the end of the examination.
10. ಪರೀಕ್ಷಾ ಅವಧಿ ಮುಗಿಯುವ ಮುನ್ನ ಪರೀಕ್ಷಾ ಕೇಂದ್ರದಿಂದ ಹೊರಗೆ ಹೋದಲ್ಲಿ ಪ್ರಶ್ನೆ ಪತ್ರಿಕೆಯನ್ನುಕೊಠಡಿಯ ಮೇಲ್ವಿಚಾರಕರಿಗೆ ಕೊಡಬೇಕು.
    Candidate is required to handover the question paper to the room invigilator in case he / she leaves the examination hall before the closing of
11. ಉತ್ತರಗಳ ಕೊನೆಗೆ) ಕೊಠಡಿಯ ಮೇಲ್ವಿಚಾ ರಬೇಕು.
    When ca shall continue to sit in the book.
12. ಉತ್ತರ ಪತ್ರ ಸಂಖ್ಯೆ, ಗುರುತುಗಳನ್ನು ಉದಾಹರಣೆಗೆ ಶ್ರೀ ಎಂದಾಗಲಿ, ಓಂ ಎಂದಾಗಲಿ, ನಮಃ ಶಿವಾಯ, ಚುಕ್ಕೆಗಳು ಇತ್ಯಾದಿಗಳನ್ನು ಬರೆಯಬಾರದು.
    Code marks revealing identity such as "Om", "Shri", "Namah-Shivaya" etc., should not be written any where in the answer books.
13. ಮೊಬೈಲ್ ಫೋನ್‌ಗಳನ್ನು ಹಾಗೂ ಇತರೆ ಎಲೆಕ್ಟ್ರಾನಿಕ್ಸ್ ಉಪಕರಣಗಳನ್ನು ಪರೀಕ್ಷಾ ಕೊಠಡಿಯೊಳಗೆ ತರಬಾರದು.
    Do not bring Mobile Phones and other electronic equipments inside the Examination Hall.
14. ಉತ್ತರ ಪತ್ರಿಕೆಯಿಂದ ಯಾವುದೇ ಪುಟವನ್ನು ಹರಿಯಬಾರದು ಅಥವಾ ತೆಗೆಯಬಾರದು.
    Do not tear or remove any sheet from the Answer Booklet

QP: 54013 Pkt : 0001 Sl.: 03
PROG. FUNDAMENT. &

Part - B

Ans 9

(a) The basic Structure of C Program is :-

1. # Include <stdio.h> :-

Preprocessor directive to be included.

2. Void Main() :-

Function name with return type data i.e., Void.

{

3. Body :-

Body of the Program or function. It may Contain in one or more statements of any type.

Statement 1;

Statement 2;

}

Every Program is treated as a function and thus there should be the function name. Here it is Main () as this is the main function in the Program. There is also the return type of data from the execution of function prefixed to it. Here nothing is returned so Void. Starting flower braces

"{" and closing flower braces "}" Makes the body of function. The body may have one or more statements. A semicolon ";" ends every statement. The body may only contain a semicolon also.

(b) Some of the basic rules need to follow while constructing flowchart are as follows:

1. All boxes of the flowchart are connected with Arrow (Not lines).
2. Flowchart symbols have an entry point on the top of the symbol with no other entry point. The exit point for all flowchart symbols is on the bottom except for the decision symbol.
3. The decision symbol has two exit points; these can be on the sides or the bottom and one side.
4. Generally a flowchart will flow from top to bottom. However, an upward flow can be shown as long as it does not exceed 3 symbols.

5. Connectors are used to connect breaks in the flowchart. Examples are:-
   - From one page to another page.
   - From the bottom of the page to the top of the same page.
6. An upward flow of more than 3 symbols.
7. Subroutine and Interrupt Programs have their own independent flowcharts.
8. All flowcharts start with a terminal or predefined process symbol.
9. All flowcharts ends with a terminal or a contentious loop.

Ans 12

(a) Bitwise operators allow manipulation of the actual bits held in each byte of a variable. Each byte consists of a sequence of 8 bits, each of which can store the value 0 or 1.

Operators	Operation
~	ones Complement
&	bitwise AND
^	bitwise XOR

|	bitwise OR
<<	left shift.
>>	Right shift

Example:-

AND	XOR	OR
0 & 0 = 0	0 ^ 0 = 0	0 | 0 = 0
0 & 1 = 0	1 ^ 0 = 1	1 | 0 = 1
1 & 0 = 0	0 ^ 1 = 1	0 | 1 = 1
1 & 1 = 1	1 ^ 1 = 0	1 | 1 = 1

(b) The different storage ~~class~~ in C are as follows:-

1. auto :-

The default class. Automatic variables are local to their block. Their storage space is reclaimed on exit from the block.

2. register :-

If possible, the variable will be stored in a processor register. May give faster

access to the variable. If register storage is not possible, then the variable will be of automatic class. Use of the register class is not recommended, as the compiler should be able to make better judgement about which variables to hold in register, In fact in judicious use of register variables may slow down the programs.

3. extern :-

Allows access to external variables. An external variables is either a global variable or a defined in another source file. External variables are defined outside of any function.

4. static :-

On exit from block, static variables are reclaimed. They keep their value on re-entry to the block the variable will have its old value.

5. [illegible]

5. Static external :-

External variable can be accessed by any function in any source file which make up the final Program. Static external variables can only be accessed by function in the same file as the variable declaration.

Ans13 #

(a)

```
#Include <stdio.h>
#Include <conio.h>
int strlen (char str[50])
{
  int len=0;
  while (str[len] != '\0')
  len++;
  len--;
  return len;
}
void strcat (char str1[50], char str2[50])
{
  int i=0, len=0
```

```
while (str2[i] != '\0')
{
  str1[len] = str2[i];
  i++;
  len++;
}
str1[len] = '\0';
}
void main()
{
  char str1[50], str2[50], str3[50], temp[50];
  int len1, len2, len3, i, j, match, k;
  clrscr();
  printf("\n\n\t Enter a sentence:");
  gets(str1);
  len1 = strlen(str1);
  printf("\n\n\t Enter a string which you want
  to delete:");
  get(str2);
  len2 = strlen(str2);
  printf("\n\n\t Enter a string which you want to
  insert:");
  get(str3);
```

```
len3 = strlen(str3);
for (i=0; i<=len1; i++)
{
  match = 1;
  for (j=0; j<= len2; j++)
  if (str2[j] != str1[i+j])
  {
    match = 0;
    break;
  }
  if (match)
  {
    for (k=0, j=i+len2+1; j<=len1; j++,
    k++)
    temp[k] = str1[j];
    temp[k] = '\0';
    for (j=0; j<= len3; j++)
    str1[i+j] = str3[j];
    str1[i+j] = '\0';
    strcat(str1, temp);
    len1 = len1 - len2 + len3;
    j = i+j;
  }
}
```

```
printf("\n\n\t Output is:");
puts(str1);
getch();
}
```

(b) `#define LIMIT 100`
`#define PI 3.14159`
`#define NAME "Steve"`

The Preprocessor allows the creation of symbolic constants.

`#define LIMIT`

By convention the symbol is given all upper case identifiers.

100 an Int Constant.

3.14159 a doble constant.

"Steve" a string Constant.

the Preprocessor will replace all tokens which have been # defined by their definition, e.g.

```
printf("%f\n", 2*PI*radius);
```

becomes

```
printf("%f\n", 2*3.14159*radius);
```

but

```
printf("%s\n", NAME);
```

does not become
Printf ("%.1f", "Steve" S);

Ans 15

(a)

```
#include <stdio.h>
Struct Employee
{
  Char ename [20];
  Int SSn;
  float salary;
  Struct date
  {
    Int date;
    Int month;
    Int year;
  } doj;
};
Emp = {"Neil", 1000, 16000.500, 8, 5, 2015};
```

```
emp = {"Neil", 1000, 16000.500, {28,5,2015}};
int main(int argc, char *argv[])
{
    printf("\n Employee name: %s", emp.ename);
    printf("\n Employee SSn : %d", emp.SSn);
    printf("\n Employee Salary: %f", emp.salary);
    printf("\n Employee DOJ : %d/%d/%d", emp.doj.date,
    emp.doj.month, emp.doj.year);
    return 0;
}
```

(b)

```
enum day {sun, mon, Tue, Weds, Thurs, Fri, Sat} d1, d2;
enum suit {Spades, Heart, Clubs, diamonds} s1;
enum suit s2, s3;
```

The Identifiers in the enumerated type list can be used in assignments or tests.

Each value of enumerated type list is given an int value. If no extra information is provided by the programmer, the values start at zero and increase by one from left to right. The value of any constants in the enumeration list can be set by the programmer, subsequent enumeration constants

will be given values starting from this value. There is no need for the values given to the enumeration constants to be unique. The following code fragments illustrates these points:-

enum days {sun=1, mon, tues, weds, thurs, fri=1, sat} d1;

The values of enumeration constants will be:

sun 1, mon 2, tues 3, weds 4, thurs 5, fri 1, sat 2

Enumeration constants must be unique across all enumerated types currently in scope, the following would be illegal:

enum day {sun, mon, tues, weds, thurs, fri, sat};
enum weekend {sat, sun} d2;

The base type of an enumerated type is int and values of type int can be assigned to variables of an enumerated type, although such use may be meaningless, e.g.,

enum days {sun, mon, tues, weds, thurs, fri, sat} d1 = 76;

5 Some debuggers are able to show the values of variables of enumerated type using the identifiers used in the enumeration list, this helps with debugging.

Part-A

Ans2
```
#include <stdio.h>
main()
{
  int n, c, first = 0, second = 1, next;
  Cout << "Enter the number of terms of Fibonacci series you want" << endl ; Cin >>n;
  Cout << " First " << n << " terms of Fibonacci series are :- " << endl ;
  for (c=0 ; c<n; c++)
  {
   if (c<=1)
   {
     next = c;
   }
   else
```

```
    {
        next = first + second;
        first = second;
        second = next;
    }
    cout << next << endl;
}
return 0;
}
```

Ans3

```
#include <stdio.h>
#include <conio.h>
void main()
{
    float c,f;
    clrscr();
    printf("\nEnter the temperature in degree:");
    scanf("%f", &c);
    f = 1.8*c + 32;
    printf("\n%f Degree = %f Fahrenheit",
        c,f);
    printf("\n\n Enter the temperature in fahrenheit");
    scanf("%f", &f);
```

```
C = (F-32)/1.8;
printf("\n\t %2f fahrenheit = %.2f degree",
    f, c)
getch();
}
```

Ans 7 The use of Prototype or the new style function definition allows the compiler to check that the parameters to a function are sensible, as well as checking the return type of the function but only if the definition or Prototype appears, before the function call.

Ans 5 All of the structure discussed up to this point have elements that can only be refrenced using fixed field names. Another means of accessing structures is to use dynamic field names. These names express the field as a variables expression that MATLAB evaluates at run-time. The dot Parenth-esis syntax shown here makes expression a dynamic field name.

Part - B

Ans 14

(b) To declare a Pointer variable we must state the type of object that the Pointer will point at. To declare a Pointer to an int, we write:-

```
int *Ptr;
```

which says that the type of *Ptr is int, that is the object that we find when following the Pointer Ptr is an integer. Therefore the type of Ptr is Pointer to int, or int *.

When declaring several pointers in a comma separated list, it is necessary to place a * before the identifier of each Pointer.

```
int *Ptr1, *Ptr2, *Ptr3;
```

this declare three Pointers to Integer, whereas

```
int *Ptr1, Ptr2, Ptr3;
```

declare one Pointer Variable and two Integer Variables.

When using initialisers with Pointer variables, it is the variable itself that is being initialised and not anything at the other end of the Pointer.

```
main()
{
    int i=17;
    int *pi=&i;
    *pi=25;
    printf("%d", *pi);
    printf("%d", pi);
    return 0;
}
```

3

Part - A

Ans 6 Constant is a identifier with an associated value which cannot be altered by the program during normal execution. The value is constant.

1

Ans A variable of union type may hold object of different types and sizes, the object all occupying the same area of storage.

Ans A Programming language that is once removed from a Computer Machine language. Machine language consist entirely of numbers and are almost impossible for humans to read and write. Assembly languages have the same structure and set of commands as Machine language but they enable a Programmer to use names instead of numbers.