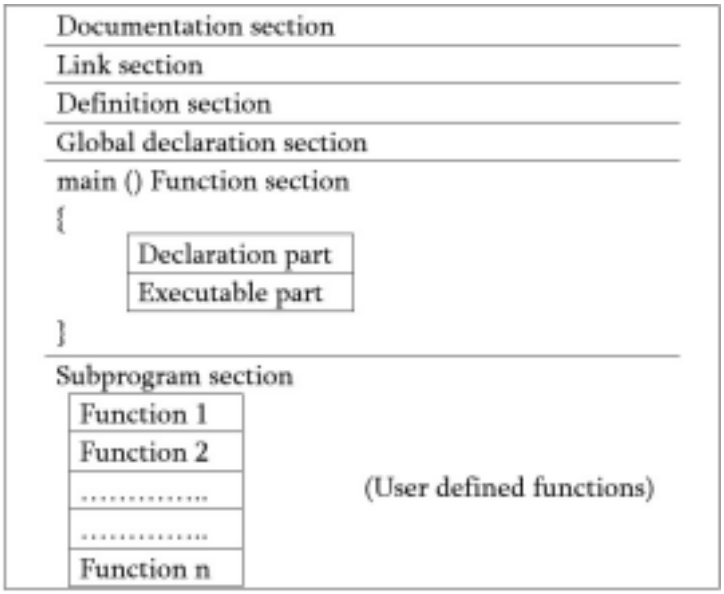
Sol with scheme-Model Answer

Prof. Rajeshwari R,Assistant Professor(CSE)

USN											CMRIT			
Internal Assessment Test 1 – April 2024														
Sub:	**Principles of Programming Using C/ Introduction to C Programming**									Sub Code	BPOPS203/ BESCK204E	Branch	**CSE, CSE(DS), ECE**	
Date:	**12.04.2024**	Duration:	90 mins	Max Marks						50	Sem / Sec:	**II Sem P-Cycle (I,J,K,L,M,N,O,P)**		
											OBE			
	Answer any FIVE FULL Questions									MARKS	CO	RBT		
1.	Explain structure of a C program with example program.									[10]	CO3	L2		
2.	With a neat diagram explain the steps in the execution of C program.									[10]	CO5	L2		
3.	Create a flowchart, develop an algorithm and write a C program to calculate both simple and compound interest.									[10]	CO5	L3		
4. a.	Classify any five input and output devices commonly used with computers..									[6]	CO1	L2		
b.	Write equivalent expressions in C: $\frac{-b+\sqrt{b^2-4ac}}{2a}$ **(2M)**													
c.	Write the output for the followings: (1) int x=8; int y=7; x++; x+=y--; (2) int a = 5, b = 10;float result; result = (float)a / b; print("Result: %0.2f\n",result); **1M(1&2)** (3) int a=100,b=200,c=300; print("%6d%5d%3d",a,b,c); (4)print("%0.3f",20/3.0); **1M(3&4)**									[4]	CO2	L3		
5.	Explain the following operators with example (a)Ternary operator (b) Unary operator									[10]	CO2	L2		
6.	Explain the syntax with examples of print() and scanf() statements. How string with blanks can be accepted by gets() and scanf() statements? Write syntax with examples.									[10]	CO2	L2		
7. a.	Illustrate the syntax for nested-if statement and apply it to write a C code to find largest of three numbers.									[6]	CO2	L3		
b.	Differentiate while and do-while statements with proper syntax and example.									[4]	CO2	L2		
8. a.	Write a C program to read maths, English, kannada subject marks and find total, average and percentage and print.									[5]	CO2	L3		
b.	Explain the break and continue statements with examples for each.									[5]	CO2	L2		

1	Explain structure of a C program with example program. Ans: **Strucure [4m]**	[10]



Definition [3M]

· **Documentation Section:** This section is used to write Problem, file name, developer, date etc in comment lines within /*….*/
or separate line comments may start with // . Compiler ignores
this section. Documentation enhances the readability of a
program.

· **Link section :** To include header and library files whose in-built functions are to be used. Linker also required these files to build
a program executable. Files are included with directive #
include

· **Definition section:** To define macros and symbolic constants by preprocessor directive #define

· **Global section:** to declare global variables – to be accessed by all functions

· **main() i**s the user defined function which is recognized by the compiler first. So, all C program must have user defined
function main() { ……. }. It should have declaration part first
then executable part.

· **Sub program section:** There may be other user defined functions to perform specific task when called.

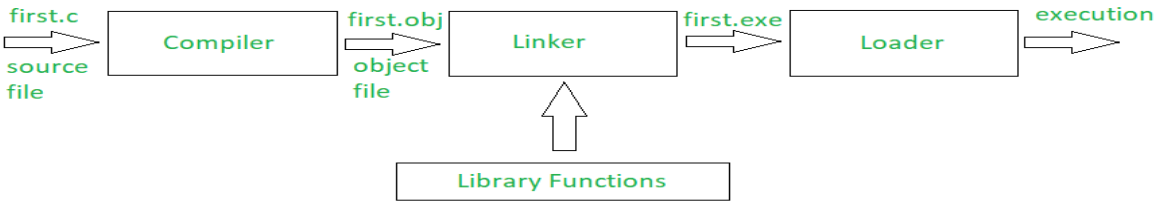
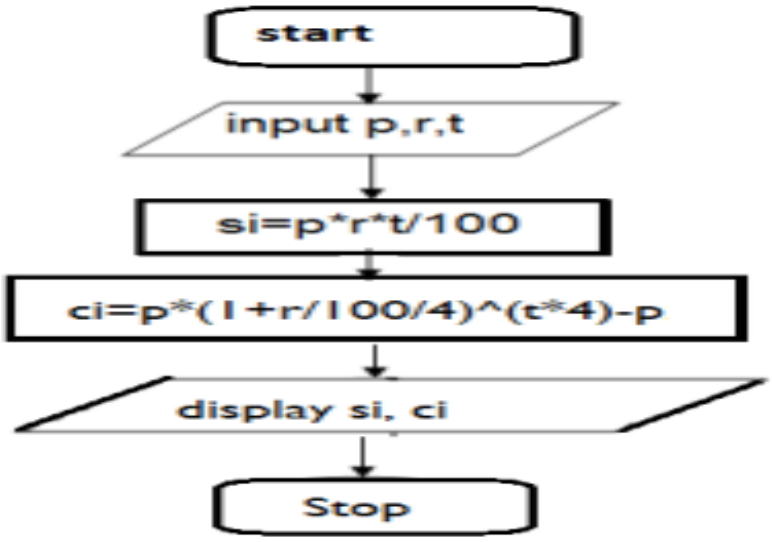
/* Example: a program to find area of a circle – area.c **[3M]**
- Documentation Section*/

```
#include <stdio.h> /* - Link/Header Section */

#define PI 3.14 /* definition/global section*/

int main() /* main function section */
{
 float r, area; /* declaration part */

 print("Enter radius of the circle : "); /* Execution part*/
 scanf("%f", &r);
 area=PI*r*r; /* using symbolic constant PI */
 print("Area of circle = %0.3f square unit\n", area);
 return (0);
}
```

2.	With a neat diagram explain the steps in the execution of C program. Ans: A compiler converts a C program into an executable. There are four phases for a C program to become an executable: **Defintion[2M]** ● Pre-processing ● Compilation ● Assembly ● Linking **Diagram[2M]**  1. Pre-processing **Explanation[6M]** This is the first phase through which source code is passed. This phase includes: ● Removal of Comments ● Expansion of Macros ● Expansion of the included files ● Conditional compilation The preprocessed output is stored in the filename.I 2. Compiling The next step is to compile filename.i and produce an; intermediate compiled output file filename.s. This file is in assembly-level instructions. 3. Assembling In this phase the filename.s is taken as input and turned into filename.o by the assembler. This file contains machine-level instructions. At this phase, only existing code is converted into machine language. 4. Linking This is the final phase is used to the creation of a single executable file from multiple object files.(.exe) which is ready to execute the file.	[10]
3	Draw flowchart,develop an algorithm and write a C program to calculate both simple and compound interest. Flowchart: **[3M]**  	[10]

ALGORITHM: **[3M]**

Step 1: Start
Step 2: Declare variables: p (principle), r (rate), t (terms), si (simple interest), ci (compound interest).
Step 3: Print "Enter principle, rate & terms : ".
Step 4: Read values for p, r, and t using scanf().
Step 5: Calculate simple interest (si) using the formula: si = p * r * t / 100.
Step 6: Calculate compound interest (ci) using the formula: ci = p * pow((1 + r / 100), t) - p.
Step 7: Print "Simple interest = %.2f Compound interest = %.2f\n", si, ci.
Step 8: End.

/*Computing Simple & Compound interest*/ **[4M]**

```
#include <stdio.h>
#include <math.h>
int main()
{
        float p,r,t,si,ci;
        print("Enter principle, rate & terms : ");
        scanf("%f%f%f", &p,&r,&t);
        si=p*r*t/100;
        ci=p*pow((1+r/100), t)-p;
        print("Simple interest = %.2f Compound interest = %.2f\n", si, ci);
        return (0);
}
```

4 (a) Classify any 5 input and 5 output devices commonly used with computers. **[3+3M]** [6]

Explain important input/output devices of computer.

Ans:

Input devices:

- Any hardware component that allows the user to enter data, programs, commands, and user response to a computer

Keyboard, Point & Draw

Output Devices:

- Any component capable of conveying information to a user

Monitors, Printers, Projectors

Definitions

Input devices:

Keyboard: Keys for letters of alphabet, numbers, 12 Function keys, Arrow Keys, toggle keys, Additional keys, Status lights, numkey pad **Point & Draw:**

Mouse – 2/3 buttons & a ball underneath (Ball underneath detects movements)

· Mouse actions: point, single click, double click, drag and release · Send signals to computer

· Instructor the mouse pointer on the screen to move accordingly **Other point and draw device:**

Trackball, Joystick Digitizer, Tablet and Pen Trackpad, Trackpoint, scanners, digital cameras, audio and video input

Output devices:

Monitors/Display devices : visually displays soft copies of text, graphics, and video information

Printers: Print with portrait or landscape, Impact Printers (striking against ink ribbon – dot matrix) or Nonimpact Printer (Ink-jet, Laser printers – high speed, high quality)

	Other output devices: ● LCD Panels ● LCD Projectors ● Voice-Response Systems ● Multimedia Projectors ● SpeakersHeadsets ● Fax Machine ● POS (Point-on-sale) terminals Some block diagrams	
4(b)	Write equivalent expressions in C: **[2M]** $\frac{-b+\sqrt{b^2-4ac}}{2a}$ **Ans:** (-b+sqrt(b*b-4*a*c))/(2*a)	[2]
4(c-1)	Write the output for the followings: **[0.5M]** int x=8; int y=7; x++; x+=y--; Ans: **Output:16** x = 8, y = 7 x++ makes x = 9 (as ++ is adding 1 to the variable) x += y-- gives x = x + y-- = 9 + 7 = 16. Here 'y' value is taken as 7 because y-- is a postfix operation, so the current value of y is used for the calculation and is then decremented.	[1]

4(c-2)	int a = 5, b = 10;float result; **[0.5M]**
	result = (float)a / b;
	printf("Result: %0.2f\n", result);
	Ans: **Output: Result:0.50**
	(float)a: Casting a to a float results in 5.0.
	5.0 / b: Dividing 5.0 by 10 results in 0.5.
	result stores the value 0.5.
	printf("Result: %0.2f\n", result); prints the value of result with two decimal places.	

4(c-3)	Write the output for the followings; int a=100,b=200,c=300; print("%6d%5d%3d",a,b,c); **[0.5M]** Ans: (100 within 6 space 200 within 5 space 300 within 3 space right aligned **100 200300**	[1]
4(c-4)	print("%0.3f",20/3.0); **[0.5M]** Ans: 6.667 20/3.0 performs floating-point division, resulting in approximately 6.667. %0.3f specifies to print the floating-point number with three decimal places.	
5	Explain the following operators with example a) Ternary operator Def&Exp **[5M]** Ternary Operator(Conditional if): ? : Syntax: <condtion> ? <statement for true> : < statement for false> int salary = 50000; Example1: bonus = salary > 40000 ? 10000 : 20000;	[10]

Here salary is greater than 40000 so, bonus will be 20000

```
#include <stdio.h>
int main() {
   int number;
   printf("Enter an integer: ");
   scanf("%d", &number);
   // Using ternary operator to check if number is even or odd
   (number % 2 == 0) ? printf("%d is even.\n", number) : printf("%d is odd.\n",
number);
   return 0;
}
```

Output:

```
Enter an integer: 7
7 is odd.
Enter an integer: 10
10 is even.
```

b) Unary operator

Def&Exp **[5M]**

Unary operators in C are operators that operate on a single operand. These operators are used to perform various operations such as incrementing, decrementing on a single operand.

Here are some common unary operators in C:

Increment (++): Increases the value of the operand by 1. Example: x++ or ++x

Decrement (--): Decreases the value of the operand by 1. Example: x-- or --x

The ++ operator increases the value of the operand by 1.

It can be used in two ways: pre-increment and post-increment.

Pre-increment: The ++ operator is placed before the operand.

```
++x;
```

Here, the value of x is incremented by 1 before its value is used in any expression.

Post-increment: The ++ operator is placed after the operand.

```
x++;
```

Here, the value of x is incremented by 1 after its current value is used in any expression.

Example:

```
int x = 5;int y = ++x; // y will be 6, x will be 6
```

Decrement (--):

The -- operator decreases the value of the operand by 1.

Similar to the increment operator, it can be used in two ways: pre-decrement and post-decrement.

Pre-decrement: The -- operator is placed before the operand.

```
--x;
```

Here, the value of x is decremented by 1 before its value is used in any expression.

Post-decrement: The -- operator is placed after the operand.

```
x--;
```

Here, the value of x is decremented by 1 after its current value is used in any expression.

Example:

```
int x = 5;int y = x--; // y will be 5, x will be 4
```

These operators are commonly used in loops, such as for loops, to control

iteration, and in other situations where the value of a variable needs to be incremented or decremented.

```
#include <stdio.h>
int main() {
   int x = 5, y;
   // Post-increment
   printf("Post-increment: x=%d, y=%d\n", x, y = x++);
   printf("After post-increment: x=%d, y=%d\n", x, y);
   // Pre-increment
   x = 5;
   printf("Pre-increment: x=%d, y=%d\n", x, y = ++x);
   printf("After pre-increment: x=%d, y=%d\n", x, y);
   // Post-decrement
   x = 5;
   printf("Post-decrement: x=%d, y=%d\n", x, y = x--);
   printf("After post-decrement: x=%d, y=%d\n", x, y);
   // Pre-decrement
   x = 5;
   printf("Pre-decrement: x=%d, y=%d\n", x, y = --x);
   printf("After pre-decrement: x=%d, y=%d\n", x, y);
   return 0;
}
```

Output:

```
Post-increment: x=5, y=5
After post-increment: x=6, y=5
Pre-increment: x=5, y=6
After pre-increment: x=6, y=6
Post-decrement: x=5, y=5
After post-decrement: x=4, y=5
Pre-decrement: x=5, y=4
After pre-decrement: x=4, y=4
```

6 Explain the syntax with examples of printf() and scanf() statements. How string with blanks can be accepted by gets() and scanf() statements? Write syntax with examples? [10]

Ans: [**4+4+2M**]

printf() – Library function for formatted output:

Output data can be written on to a standard output device using the library function printf(). The print statement provides certain features that can be effectively exploited to control the alignment and spacing of printouts on the terminals. The general form of print statement is:

printf ("control string", arg1,arg2, ..., argn);

scanf() – Library function for formatted input Formatted input refers to an input data that has been arranged in a particular format. Input data can be entered into the computer from a standard input device by means of the C library function scanf. In general terms, scanf function is written as

scanf ("control string", &arg1, &arg2, ..., &argn);

The control string specifies the field format in which the data is to be entered and the arguments arg1,arg2.....,argn specify the address of locations where the data is stored. when scanf() is used to read string input it stops reading when it encounters

	whitespace, newline or End Of File
	scanf (“% [^\n]”, str);
	Can be used to read string with blanks
	Example:
	#include <stdio.h>
	int main() {
	char name_gets[20];
	char name_scanf[20];
	// Using gets() to accept string with blanks
	printf("Enter name using gets(): ");
	gets(name_gets); // Avoid using gets() due to security vulnerabilities, but used here for demonstration
	puts(name_gets);
	// Using scanf() to accept string with blanks
	printf("Enter name using scanf(): ");
	scanf(" %[^\n]", name_scanf);
	printf("%s\n", name_scanf);
	return 0;
	}
	Explanation:
	gets(name_gets): This function reads a line of text from standard input and stores it in the name_gets array. It reads characters until a newline character or EOF is encountered, including any spaces or tabs.
	scanf(" %[^\n]", name_scanf): This scanf() call reads a string until a newline character (\n) is encountered, including any whitespace characters like spaces or tabs. The leading space before %[^\n] is essential to consume any leading whitespace characters leftover from previous input operations.
	Output:
	Enter name using gets(): John Doe
	John Doe
	Enter name using scanf(): Jane Smith
	Jane Smith	

7(a)	Illustrate the syntax for nested-if statement and apply it to write a C code to find largest of three numbers. **[2+4M]**
	Ans:
	Nested if is if within if.
	Syntax:
	if (condition) {
	if (condition) {
	statements;
	}
	else {
	statements;
	}
	}
	else {
	if (condition) {
	statements;
	}
	else {
		[10]

```
  statements;
  }
  }
program to  find largest of three numbers using nested- if statements
#include <stdio.h>
int main() {
   int a, b, c;
   printf("Enter three numbers separated by spaces: ");
   scanf("%d %d %d", &a, &b, &c);
   if (a >= b) {
      if (a >= c) {
         printf("A=%d is the largest.\n", a);
      } else {
         printf("C=%d is the largest.\n", c);
      }
   } else {
      if (b >= c) {
         printf("B=%d is the largest.\n", b);
      } else {
         printf("C=%d is the largest.\n", c);
      }
   }
   return 0;
}
```

Output:

```
Enter three numbers separated by spaces: 5 12 8
B=12 is the largest.
Enter three numbers separated by spaces: 15 8 10
A=15 is the largest.
Enter three numbers separated by spaces: 8 10 15
C=15 is the largest.
```

7(b) Differentiate while and do-while statements with proper syntax and example. **[4M]**

While Loop	**Do-While Loop**
In the While loop, the condition is tested before any statement is executed.	In Do while-loop, the statement is executed at least once even if the condition is false

Syntax:
while(condition){
// statements
}
	Syntax:
	do{
	//statements
	}while(expression);

In While loop, no semicolon is needed after the end of the condition.	In Do-while loop, semicolon needed after the end of the condition
While loop is an entry-controlled loop.	Do-while loop is an exit-controlled loop.

While loop may or may not be executed at all.	Do-while loop will execute at least once.	
While loop can lead to errors if the condition is always false.	Do-while loop help prevents error as it runs at least once.	

Example:
#include <stdio.h>
 	
int main()
{
 	
int i = 5;
 	
while (i < 10) {
printf("GFG\n");
i++;
}
 	
return 0;
}
Output:
GFG
GFG
GFG
GFG
GFG
	Example:
	#include <stdio.h>
	
	int main()
	{
	
	int i = 5;
	
	do {
	printf("GFG\n");
	i++;
	} while (i < 10);
	
	return 0;
	}
	Output:
	GFG
	GFG
	GFG
	GFG
	GFG
		[10]

8(a)

Write a C program to read Maths, English, Kannada subject marks and find total, average and percentage and print. **[Correctness-2,logic-2,output-1]**

Ans:

```
#include <stdio.h>
int main(){
 int math,eng,kan,total;
 float avg;
 print("Enter marks in 100 for Math, Eng and Kannada : ");
scanf("%d%d%d",&math,&eng,&kan);
 total=math+eng+kan;
 avg=total/3.0;
 print("Total = %d average and percentage = %.2f\n", total,avg);  return (0);
}
```

Expected output:

Enter marks in 100 for Math, Eng and Kannada : 50 60 70

Total = 180 average and percentage = 60.00

8(b) Explain the break and continue statements with examples for each. **[5M]**

Ans:

Break Statement	**Continue Statement**
The Break statement is used to exit from the loop constructs.	The continue statement is not used to exit from the loop constructs.
The break statement is usually used with the switch statement, and it can also use it within the while loop, do-while loop, or the for-loop.	The continue statement is not used with the switch statement, but it can be used within the while loop, do-while loop, or for-loop.
When a break statement is encountered then the control is exited from the loop construct immediately.	When the continue statement is encountered then the control automatically passed from the beginning of the loop statement.
Syntax: break;	Syntax: continue;
Break statements uses switch and label statements.	It does not use switch and label statements.
Leftover iterations are not executed after the break statement.	Leftover iterations can be executed even if the continue keyword appears in a loop.

Example:
#include<stdio.h>
void main()
{
int i;
for (i = 0; i < 10; i++)
{
if (i == 4)
{
printf("Skip");
break;
}
printf("%d\n", i);
}
}
Output:
0
1
2
3
Skip
	Example:
	#include<stdio.h>
	void main()
	{
	int i;
	for (i = 0; i <=5; i++)
	{
	if (i == 2)
	{
	printf("Continue next step\n");
	continue;
	}
	printf("%d\n", i);
	}
	}
	Output:
	0
	1
	Continue next step
	3
	4
	5