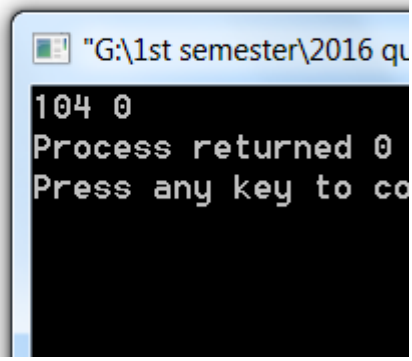
Dept. Computer Science and Engineering
University of Rajshahi
Semester Final Examination, B.Sc. Engg. 2016, 1st year, Odd semester,
Course ID: CSE -1121 Course Title: Computer Programming with C
Total Time 3 Hours Total Marks 52.5
Answer any **six** questions taking **three** from each section

Section A

1(a) If int i = 7, float f = 5.5, char c = 'a'; What will the output of (a) 'i + c' and (b) 'i + f' 3
If int result, i = 7, f = 8.5, What will the output of ' result = (i + f) % 4'
If float num = 10.5, What will the output of 'num % 2' and '((int)num) % 2'

1a.c

```c
int main(){

    int i=7;
    float f=5.5;
    char c='a';

    printf("%d ", i+c);
    printf("%d ",i+f);
    return 0;
}
```



1aii.c

```c
#include<stdio.h>

int main(){

    int i=7;
    float f=8.5;
    int result = (i+f)%4;
    printf("%d ", result);
    return 0;
}
```

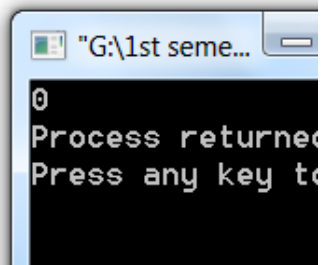
The summation of int type and float type data i+f is a floating point number. We cannot find modulus of a floating point number. The % operator cannot be applied to floating-point numbers i.e float or double. If you try to use the modulo operator with floating-point constants or variables, the compiler will produce a error.

1aiii.c

```c
#include<stdio.h>

int main(){

    float num= 10.5;
    printf("%d ", (num%2));
//    printf("%d ", ((int)num)%2);
    return 0;
}
```

1aiii.c

```c
#include<stdio.h>

int main(){

    float num= 10.5;
//    printf("%d ", (num%2));
    printf("%d ", ((int)num)%2);
    return 0;
}
```



10%2

2)10(5
10
0

(b) What will be simplified form of (a) !(a<b) , (b) !(c<=d), (c) !(x=>y) ? 1.5

b) Answer: (a) !(a<b) : a>=b (b) !(c<=d) : c > d (c) !(x=>y) : x < y

(c) What will be the output of the following code? 4.25
(Objective of the question: To check the formatting knowledge)

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

main()
{
    printf("%7d\n",123);
    printf("%-4d\n",123);
    printf("%07d\n",15);
    printf("%4.3f\n",3.14159);
    printf("%x\n",127);
    printf("%o\n",127);
    getch();
}
```

1c.c

```
2   #include<conio.h>
3   main(){
4
5       printf("%7d\n",123);
6       printf("%-4d\n",123);
7       printf("%07d\n",15);
8       printf("%4.3f\n",3.14159);
9       printf("%x\n",127);
10      printf("%o\n",127);
11      getch();
12  }
```

"G:\1st semester\2

```
    123
123
0000015
3.142
7f
177
```

Output:

				1	2	3
1	2	3				
0	0	0	0	0	1	5
3	.	1	4	2		
7	f					
1	7	7				

2(a) What will be the output of the following code? 2.75

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

int i,j;
main()

{
 i=1;
 while(i<=5)
 {
 for (j=1; j<=6; j++)
   {if (i==j)
    printf("X");
   else
    printf("Y");
    }
   i=i+1;
   printf("\n");
}

}
```

Output:

```
XYYYYY
YXYYYY
YYXYYY
YYYXYY
YYYYXY
```

2a.c

```
1   #include<stdio.h>
2   #include<conio.h>
3
4   int i,j;
5   main()
6   {
7       i=1;
8       while(i<=5){
9           for(j=1;j<=6;j++){
10              if(i==j)
11                  printf("X");
12              else
13                  printf("Y");
14          }
15          i=i+1;
16          printf("\n");
17      }
18  }
```

(b) List the syntax error (if any) of each line of the following code? 6
(Objective of the question: To check the knowledge of basic C syntax)

```
#include <conio.h>
int 1x,2x, y1,y2;
float z;
char a[10], b[10];
Main()
{
scanf("%d%d%f",y1,z);
scanf("%c%c%c), &a[1],a[2],&a[3]);
b[2]=a[2];


y2=b[2]+a[1]+y1;
printf("%f%f%f%d%d",&y1,z,y2,z,a[3]);
}
```

2b.c

```
1   #include<stdio.h>
2   #include<conio.h>
3
4   int x1, x2, y1, y2;
5   float z;
6   char a[10],b[10];
7   main()
8   {
9   scanf("%d%f",&y1,&z);
10  scanf("%c%c%c",&a[1],&a[2],&a[3]);
11  b[2]=a[2];
12  y2 = b[2]+a[1]+y1;
13  printf("%d%f%d%f%c",y1,z,y2,z,a[3]);
14  }
```

3(a) What is the difference between 'while' and 'do-while' loops? 2 6

BASIS FOR COMPARISON	WHILE	DO-WHILE

General Form	while (condition) {
	statements; //body of loop
	}
		do{
		.
		statements; // body of loop.
		.
		} while(Condition);

Controlling Condition	In 'while' loop the controlling condition appears at the start of the loop.	In 'do-while' loop the controlling condition appears at the end of the loop.
Iterations	The iterations do not occur if, the condition at the first iteration, appears false.	The iteration occurs at least once even if the condition is false at the first iteration.
Alternate name	Entry-controlled loop	Exit-controlled loop
Semi-colon	Not used	Used at the end of the loop

(b) What will be the output of the following program?
(Objective of the problem: The check the capacity of four-layer nested loop control

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

int x[5][5]={   {1, 4, 3, 6, 8},
                {2, 9, 0, 5, 7},
                {5, 9, 6, 7, 6},
                {9, 0, 2, 6, 8},
                {3, 6, 0, 1, 7} };

int i,j,k,l, tmp,big,p;

main() {

 for (i=0;i<=4;i++)
 {
   for(j=0; j<=4; j++)
   {
     for(k=j; k<=4; k++)
     {
       for(l=k ; l<=4; l++)
       {
       x[k][l]=x[k][l]+1;
       }
     }

   }
 }
```

```
for (i=0;i<=4;i++)
 {
   for(j=0; j<=4; j++)
   {
     printf ("%d ",x[i][j]);
   }
   printf ("\n ");
 }

getch();
}
```

3b.c

```
1  #include<stdio.h>
2  #include<conio.h>
3
4  int x[5][5]= {{1,4,3,6,8},
5                {2,9,0,5,7},
6                {5,9,6,7,6},
7                {9,0,2,6,8},
8                {3,6,0,1,7}};
9
10 int i,j,k,l,tmp,big,p;
11
12 main(){
13
14     for(i<0;i<=4;i++){
15         for(j<0;j<=4;j++){
16             for(k=j;k<=4;k++){
17                 for(l=k;l<=4;l++){
18                     x[k][l] = x[k][l]+1;
19                 }
20             }
21         }
22     }
23
24     for(i=0;i<=4;i++){
25         for(j=0;j<=4;j++){
26             printf("%d ",x[i][j]);
27         }
28         printf("\n");
29     }
30
31     getch();
32 }
```

"G:\...

```
2 5 4 7 9
2 11 2 7 9
5 9 9 10 9
9 0 2 10 12
3 6 0 1 12
```

4(a) What is the purpose of the switch statement? How does this statement differ from the other statements? 2.5

4 a) Answer: A **switch** statement allows a variable to be tested for equality against a list of values. Each value is called a case, and the variable being switched on is checked for each **switch case**.

```
switch(expression) {
  case constant-expression  :
    statement(s);
     break; /* optional */
  /* you can have any number of case statements */
  default : /* Optional */
  statement(s);
}
```

1. SWITCH statement is easier to express for lengthy conditions when compared to an IF statement which gets more complex as the number of conditions grow and the nested IF comes into play.

2. SWITCH statement allows easy proofreading while testing and removing bugs from the source code whereas IF statement makes editing difficult.

3. Expression is evaluated and SWITCH statement is run according to the result of the expression that can be integer or logical while IF statement is run only if the result of the expression is true.

4. SWITCH allows expression to have integer based evaluation while IF statement allows both integer and character based evaluation.

5. SWITCH statement can be executed with all cases if the 'break' statement is not used whereas IF statement has to be true to be executed further..

4(a) What is the purpose of the switch statement? How does this statement differ from the other statements? 2.5
(b) What is the difference between the break and continue statement? 2.5

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.

Difference between break and continue is as shown below:

Sr. No.	Break	Continue
1.	Break is used to break loop or iteration.	Continue continues the loop or iteration.
2.	Used with switch case loop.	Not used with switch case.
3.	Keyword used is "break".	Keyword used is "continue".
4.	Breaks loops and allows coming out from it.	Allows iterating in the loop.
5.	Control is transferred outside the loop.	Control remains in the same loop

(b) What is the difference between the break and continue statement? 2.5

(c) Write a fragment of program that makes use of the goto statement. 3.75

```
#include <stdio.h>
int main()
{
  int sum=0;
  for(int i = 0; i<=10; i++){
        sum = sum+i;
        if(i==5){
          goto addition;
        }
  }
  addition:
  printf("%d", sum);

  return 0;
}
```

Output: 15

Query: 18115@imperial.edu.bd