Answers for mid sem 1 exam [BCSC0001]

Section A

Question 1

Fill in the blanks in the following program.

```c
1  #include <stdio.h>
2  void main()
3  {
4      ____ j;
5      ____ k;
6      ____ ("ENTER THE CHARACTER & FLOAT VALUES RESP.:");
7      scanf("%c ____ ", &j, &k);
8      printf("\n%c\n%f\n", j, k);
9  }
```

Answer 1

```c
1  #include <stdio.h>
2  void main()
3  {
4      char j;
5      float k;
6      printf("ENTER THE CHARACTER & FLOAT VALUES RESP.:");
7      scanf("%c%f", &j, &k);
8      printf("\n%c\n%f\n", j, k);
9  }
```

> Explanation
>
> - Blank 1 : The variable `j` is scanned as a character in the `scanf()` function on line 7, hence it would be of `char` type.
> - Blank 2 : The variable `k` would be a `float` as the description in the `printf()` function on line 6 states that a character variable and a float variable must be entered. Since we already have a `char` type variable name `j`, then it only makes sense that the remaining variable `k` be the `float` variable.
> - Blank 3 : The syntax of the function matches to the syntax of the `printf()` function in C. We can confirm this by looking at the `String` type argument provided to the function, which is `"ENTER THE CHARACTER & FLOAT VALUES RESP.:"`.
> - Blank 4 : The `scanf()` function has to scan two variables, one `char` and one `float` as we have already discussed. Since the `char` type format specifier (`%c`) is already defined in the program, we only have to write the format specifier for `float` data type, i.e `%f`.

Question 2

Fill in the blanks in the following program

```c
#include<stdio.h>
#include<____>
void main()
{
    int ____, ____, ____;
    float v;
    scanf("%d%d%d", ____ u, ____ a, ____ s);
    v = pow((u * u + 2 * a * s), 1 / 2.0);
    printf("The final velocity = %f", v);
}
```

Answer 2

```c
#include<stdio.h>
#include<math.h>
void main()
{
    int u, a, s;
    float v;
    scanf("%d%d%d", &u, &a, &s);
    v = pow((u * u + 2 * a * s), 1 / 2.0);
    printf("The final velocity = %f", v);
}
```

> Explanation
>
> - Blank 1 : The blank specifies a *preprocessor directive* that requires the name of a header file. In this case, `math.h`. We can confirm this by looking at the program. The program uses the `pow()` function, which is defined in the `math.h` header file. Hence, in order to use the `pow()` function inside our program, we *must* include the `math.h` header file.
> - Blank 2 : The program makes use of four variables namely, `a, s, u & v`. Since the variable `v` is already defined in the program, we only have to create the other three variables namely, `a, u & v`. Hence this blank will be used to declare an `int` variable named `a, u or s`. Either one is correct. The order in which the variables are **does not** matter. Here, we have used `u` as an option.
> - Blank 3 : For the reason stated in the explanation of blank 2, we have to declare the other variables that are left. Here, we have used `a`.
> - Blank 4 : The final variable left to be declared is `s`.

> - Blank 5 : The `scanf()` function on line 7 is is to be used for scanning three variables, namely, `u, a and s`. The syntax of the `scanf()` function dictates that the second argument must be the address of the variable in which the value is to be stored. Hence we will use the `&` operator with the name of the variable to denote the address of the variable. Answer is `&`.
> - Blank 6 : Similarly as in blank 5, the answer is `&`.
> - Blank 7 : Similarly as in blank 5 & 6, the answer is `&`.

Question 3

What is the output of the following program?

```c
#include<stdio.h>
int main()
{
    float c = 5.0;
    printf("Temperature in Fahrenheit is %.2f", (12 / 5) * c + 30);
    return 0;
}
```

Answer 3

```
Temperature in Fahrenheit is 40.00
```

> Explanation
>
> The values `12` and `5` are of *integer* type. Hence any operation done on both these values will give a result in *integer* type. So, the operation `(12 / 5)` will yield an *integer* result. The result of this operation is **integer** `2`. Since the variable `c` is of *float* type, the computer will try to convert the result of the operation `(12 / 5) * c` into a *float* type. It will use **implicit type casting** to do so. The result of this operation would be a **float** **`10.000000`**. ***Remember***, the result of these two operations has a *float* type result now and all further operation would be done with this float type result. Now, when we add an *integer* `30` to the operation `(12 + 5) * c`, the resulting value will be a **float** **`40.000000`**. The answer however, is `Temperature in Fahrenheit is 40.00` because the floating point number has the print format of at most two places after the decimal (`%.2f`).

Question 4

What is the output of the following program?

```c
#include<stdio.h>
int main()
{
    int a = 15, b = 26, c = 35;
    if (c > b > a)
        printf("Hello");
    else
        printf("Bye");
    return 0;
}
```

Answer 4

```
Bye
```

> Explanation
>
> The expression `(c > b > a)` is treated as `((c > b) > a)` by the compiler because of the associativity of the `>` operator is defined left to right. That means, it will evaluate the left-most expression first. Therefore the actual expression now becomes `((15 > 26) > 35)`. The first expression to be evaluated would be `(15 > 26)`, which would result in *false* and the compiler will yield the value `0` for *false* (otherwise `1` for true). Since the first expression has now been evaluated, we will substitute the value of the first expression back into the complete expression which will look like this `(0 > a)`. This expression will also result in *false* since `0` is not greater than `a`. So finally, the `else` block of the `if-else` block will be executed and the output will be `Bye`.

Question 5

Fill in the blanks in the following program.

```c
 1  #include<stdio.h>
 2  void main()
 3  {
 4      ____ c;
 5      scanf("%c", &c);
 6      if ((c >= 65 && c <= 90) ____ (c >= 97 && c <= 122))
 7          printf("IT IS ALPHABET");
 8      ____ ____ (c >= 48 && c <= 57)
 9          printf("IT IS DIGIT");
10      else
11          printf("IT IS SPECIAL CHARACTER");
12  }
```

Answer 5

```c
 1  #include<stdio.h>
 2  void main()
 3  {
 4      char c;
 5      scanf("%c", &c);
 6      if ((c >= 65 && c <= 90) || (c >= 97 && c <= 122))
 7          printf("IT IS ALPHABET");
 8      else if (c >= 48 && c <= 57)
 9          printf("IT IS DIGIT");
10      else
11          printf("IT IS SPECIAL CHARACTER");
12  }
```

> Explanation
>
> - Blank 1 : The variable `c` is scanned as a *character* type in the `scanf()` on line 5. From here we can confirm that the variable `c` is a `char` type variable.
> - Blank 2 : The expression written in `if()` checking the value of variable `c` inside the range `[65, 90]` **OR** `[97, 122]`. This is the range of *lowercase* and *uppercase* alphabets in the *ASCII* table. Since an alphabetic character can only be uppercase *or* lowercase at any time, we put `||` in the blank 2.
> - Blank 3 : The `if-else` block is being continued and the `char` variable `c` is now being checked whether it is present in another range `[48, 57]`. This is the range of numeric digits in the *ASCII* table. Since the variable can *only* be either a alphabet *OR* a number, this statement must be an `else`.
> - Blank 4 :

Question 6

Fill in the blanks in the following program.

```
 1  #include<stdio.h>
 2  int main()
 3  {
 4      ____ ____, ____, ____;
 5      printf("Enter first number: ");
 6      scanf("%f", &first);
 7      printf("Enter second number: ");
 8      scanf("%f", &second);
 9      ____ = first;
10      first = second;
11      ____ = third;
12      printf("\nAfter swapping, firstNumber = %f\n", first);
13      printf("\nAfter swapping, secondNumber = %f\n", second);
14      return 0;
15  }
```

Answer 6

```
 1  #include<stdio.h>
 2  int main()
 3  {
 4      float first, second, third;
 5      printf("Enter first number: ");
 6      scanf("%f", &first);
 7      printf("Enter second number: ");
 8      scanf("%f", &second);
 9      third = first;
10      first = second;
11      second = third;
12      printf("\nAfter swapping, firstNumber = %f\n", first);
13      printf("\nAfter swapping, secondNumber = %f\n", second);
14      return 0;
15  }
```

> Explanation
>
> - Blank 1 : As we can see from the program, some variables namely, `first, second and third` have been used but have not been declared. From here, we know that we must declare them. Therefore, the first line of the program must be the declaration of all the variables. The first blank then should be the data type of the variables. From the `scanf()` function on line 6, we can see that the variables are scanned as `float` variables. Hence, the answer is `float`.

> - Blank 2 : The variables we have seen in the program are `first, second and third`. The *order* in which we declare the variable does not affect the output of the program. Here, we have used `first`.
> - Blank 3 : Similarly as in blank 2, we are using `second`.
> - Blank 4 : Similarly as in blank 3, we are using `third`.
> - Blank 5 : Since this program is swapping the two numbers `first` and `second` using the third variable `third`, we will have to use the logic
>   - store the value of first variable in third variable
>   - store the value of second variable in first variable. ***Remember***, this will overwrite the value of first variable.
>   - store the value of third variable in second variable.
>
>   Thus we write `third` in this blank.
> - Blank 6 : Similarly as in blank 5, we write `second` in this blank.

Question 7

What will be the output of following program? Assume integer is of 4 bytes.

```c
#include<stdio.h>
void main()
{
    printf("%d %d", sizeof(int), sizeof(char));
}
```

Answer 7

```
4 1
```

> Explanation
>
> The `sizeof()` operator returns the size of the argument that is passed into it in *bytes*. Here, we are passing the data types `int` and `char` to the `sizeof()` operator, so the result would be the number of bytes that the data types requires to store a value. In the question itself, it is given that the size of `int` is to be assumed as `4` bytes. Since, we know that the `char` data type only stores a single character at any time, the size would be only `1` byte. Hence, the answer `4 1`.

Question 8

What will be the output?

```
#include<stdio.h>
void main()
{
    int x = 1, y = 0, z = 5;
    int a = x++ && ++y && z++;
    printf("%d %d %d", x, y, z);
}
```

Answer 8

```
2 1 6
```

> Explanation
>
> The prefix (++x) and the postfix (x++) operator both increase the value of an operand by 1 unit.
>
> Here, all of the operators are increasing the values of the operands `x`, `y`, and `z`. Since, the values of the variables are *initialized* as `1`, `0` and `5`, the values will be increased to `2`, `1` and `6`.

Question 9

What will be the output?

```
#include<stdio.h>
void main()
{
    int x = 7;
    if (x > 4)
        printf("hello");
    else if (x == 7)
        printf("hi");
    else
        printf("no");
}
```

Answer 9

```
hello
```

> Explanation
>
> The value of `x` is initialized as `7`. The first condition in the `if()` would evaluate as *true* and the block will be executed. The compiler will **NOT** evaluate the `if-else` ladder any further!

Question 10

What will be the output?

```
 1  #include<stdio.h>
 2  int main()
 3  {
 4      int a = 4;
 5      switch (a)
 6      {
 7          case 1  :   printf("First");
 8          case 2  :   printf("Second");
 9          case 2 + 2  :   printf("Third");
10          case 4  :   printf("Final");
11              break;
12      }
13      return 0;
14  }
```

Answer 10

```
|9|error: duplicate case value
|10|error: duplicate case value
```

> Explanation
>
> The `switch-case` can execute only one case if matched at a time. Since there are two possible values here that can be executed, no case would be executed because both of them are equally as eligible to be executed. This case is called ***ambiguity***. This program would produce an error stating that the case on line 9 and case on line 10 have a duplicate value.

Section B

Question 11

Fill in the blanks in the following program.

```c
 1  #include<stdio.h>
 2  void main()
 3  {
 4      int g1, g2, c1, c2;
 5      float spi;
 6      printf("Enter the grades in 2 subjects:");
 7      ____ ("____ ____", &g1, &g2);
 8      printf("Enter the credits in respective subjects:");
 9      ____ ("%d%d", ____, ____);
10      spi = (float) (g1 * c1 + g2 * c2) / (c1 + c2);
11      printf("The SPI = ____", spi);
12  }
```

Answer 11

```c
 1  #include<stdio.h>
 2  void main()
 3  {
 4      int g1, g2, c1, c2;
 5      float spi;
 6      printf("Enter the grades in 2 subjects:");
 7      scanf("%d%d", &g1, &g2);
 8      printf("Enter the credits in respective subjects:");
 9      scanf("%d%d", &c1, &c2);
10      spi = (float) (g1 * c1 + g2 * c2) / (c1 + c2);
11      printf("The SPI = %f", spi);
12  }
```

> Explanation
>
> - Blank 1 : By the syntax of the function, we can observe that there are variable names in the statement used with `&`, this means that this function is `scanf()`.
> - Blank 2 : In the declaration above, we can observe that the variables `g1` and `g2` are declared as `int` type. Therefore, we will use the `%d` format specifier here.
> - Blank 3 : Similarly as in blank 2, we user `%d`.
> - Blank 4 : Similarly as in blank 1, we will use the `scanf()` function here. Another way to confirm this is, in the previous line (line no 8), we have asked the user to enter the credits, so naturally, we would be using a `scanf()` function next.

> - Blank 5 : Since we have to enter the two values for `c1` and `c2`. We will use `&c1` and `&c2`. Here, we use `&c1`.
> - Bank 6 : Similarly, as in blank 5, we use `&c1`.
> - Blank 7 : Since we have to print the `float` variable `spi`. We will use the format specifier `%f`.

Question 12

Fill in the blanks in the following program.

```c
#include<stdio.h>
void main()
{
    ____ a, b, c;
    ____ z;
    printf("Enter the sides of triangle:");
    scanf("%f%f%f", &a, &b, &c);
    z = (c + a > b) && (b + a > c) && (b + c > a) ____ 1 : 2;
    if (z == 1)
        printf("TRIANGLE IS VALID");
    ____
        printf("TRIANGLE IS NOT VALID");
}
```

Answer 12

```c
#include<stdio.h>
void main()
{
    float a, b, c;
    int z;
    printf("Enter the sides of triangle:");
    scanf("%f%f%f", &a, &b, &c);
    z = (c + a > b) && (b + a > c) && (b + c > a) ? 1 : 2;
    if (z == 1)
        printf("TRIANGLE IS VALID");
    else
        printf("TRIANGLE IS NOT VALID");
}
```

> Explanation
>
> - Blank 1 : From line 7, we can see that the variables `a`, `b` and `c` are of `float` types. Therefore, we use `float` here.

> - Blank 2 : From line 8, we can observe that the value of `z` is either `1` or `0`, hence we can say that the variable `z` should be of `int` type.
> - Blank 3 : The syntax of the *ternary* operator is `? :`. Following this, we will write `?` here.
> - Blank 4 : The `-if-else` block is incomplete here. We will write `else`.

Question 13

Fill in the blanks in the following program.

```
#include<stdio.h>
int main()
{
    int n1, n2, n3;
    printf("Enter three numbers: ");
    scanf("%d%d%d", &n1, &n2, &n3);
    if (n1 > n2)
    {
        if (____)
            printf("%d is the largest number", n1);
        else
            printf("%d is the largest number", ____);
    }
    ____
    {
        if (n2 > n3)
            printf("%d is the largest number", ____);
        else
            printf("%d is the largest number", n3);
    }
    return 0;
}
```

Answer 13

```
#include<stdio.h>
int main()
{
    int n1, n2, n3;
    printf("Enter three numbers: ");
    scanf("%d%d%d", &n1, &n2, &n3);
    if (n1 > n2)
    {
        if (n1 > n3)
            printf("%d is the largest number", n1);
        else
```

```c
                printf("%d is the largest number", n3);
        }
        else
        {
            if (n2 > n3)
                printf("%d is the largest number", n2);
            else
                printf("%d is the largest number", n3);
        }
        return 0;
    }
```

> Explanation
>
> In this program, we are comparing three numbers and checking which number is the largest.
>
> The logic for this program is as follow
>
> - if `n1 > n2`
>   - if `n1 > n3`
>   - Then, `n1` is the largest number.
>   - else
>   - Then, `n3` is the largest number.
> - else
>   - if `n2 > n3`
>   - Then, `n2` is the largest number.
>   - else
>   - Then, `n3` is the largest number.

Question 14

What does the following algorithm do?

```
Step 1  :   Start
Step 2  :   Input a no N
Step 3  :   i = 1
Step 4  :   T = N * i
Step 5  :   print T
Step 6  :   i = i + 1
Step 7  :   Repeat steps 4 - 6 till i <= 10
Step 8  :   Stop
```

Answer 14

```
The above algorithm is for printing the multiplication table of any given number N
upto 10 terms.
```

Question 15

Calculate the values of variables b, d, e for the following code segments? Assume n = 14.

```
b = n << 2;
d = 5 ^ 3;
e = 9 & 11;
```

Answer 15

```
56
6
9
```

> Explanation
>
> - For `b = n << 2` where the value of n is `14` we have to first convert the value of decimal `14` to binary. The equivalent binary value will be `1110`.
>   - After shifting `1110` two digits to the left it will become
>
>   `110110`
>
>   which will again be converted to `int` for printing.
> - From the BINARY LOGIC, you might remember
>   - AND **(&)** is only true when both operands are true
>   - when both operands are 1, it will return 1
>   - for any other inputs, it will return 0
>   - OR **(|)**
>   - when either of the inputs is 1, it will return 1
>   - NOT **(~)**
>   - it simply converts the input 1 to 0 or 0 to 1.
>   - XOR **(^)**
>   - it only return 1 when the input pair is both 1 and 0.
> - For `d = 5 ^ 3`

> - `0101` ^ `0011` = `0110`
>   - starting from the right-most digit
>   - `1` ^ `1` = `0`
>   - `0` ^ `1` = `1`
>   - `1` ^ `0` = `1`
>   - `0` ^ `0` = `0`
> - For `e = 9 & 11`
>   - `1001` * `1011` = `1001`
>   - starting from the right-most digit
>   - `1` & `1` = `1`
>   - `0` & `1` = `0`
>   - `0` & `0` = `0`
>   - `1` & `1` = `1`

Section C

Question 16

What will be the values of the following expressions? Assume a = 3, b = 4, c = 9, d = 6.

```
p = - ++d / 2;
m = ++a * b - c % 4 + d;
```

Answer 16

```
If the expressions are considered in a single program
    p = -3
    m = 22
If the expression are considered individually
    p = -3
    m = 22
```

Question 17

What does the following program do? Assume a = 15, b = 6.

```c
#include<stdio.h>
void main()
{
    int a, b, c, r;
    printf("Enter two numbers")
    scanf("%d%d", &a, &b);
    c = a / b;
    r = a - b * c;
    printf("%d", r);
}
```

Answer 17

```
The program calculates the quotient and remainder of two numbers a and b and prints
the remainder.
```

Question 18

What does the following program do?

```c
#include<stdio.h>
void main()
{
    int n = 5467, r, s = 0;
    r = n % 10;
    s = s * 10 + r;
    n = n / 10;
    r = n % 10;
    s = s * 10 + r;
    n = n / 10;
    r = n % 10;
    s = s * 10 + r;
    n = n / 10;
    printf("%d", s);
}
```

Answer 18

```
The program reverses the last three digits of a number and prints them.
```

> Explanation
>
> n = 5467, s = 0
>
> r = n % 10;
>
> - r = 7
>
> s = s * 10 + r;
>
> - s = 0 * 10 + 7
> - s = 0 + 7
> - s = 7
>
> n = n / 10;
>
> - n = 5467 / 10
> - n = 546
>
> r = n % 10;
>
> - r = 546 % 10
> - r = 6
>
> s = s * 10 + r;
>
> - s = 7 * 10 + 6
> - s = 70 + 6
> - s = 76
>
> n = n / 10;
>
> - n = 546 / 10
> - n = 54
>
> r = n % 10;
>
> - r = 54 % 10
> - r = 4
>
> s = s * 10 + r;
>
> - s = 76 * 10 + 4
> - s = 760 + 4
> - s = 764
>
> n = n / 10;
>
> - n = 54 / 10
> - n = 5

Question 19

What will be the output of the following program?

```c
#include<stdio.h>
int main()
{
    int a = 5;
    switch(a)
    {
            default : printf("Sorry input again");
            case 1 : printf("How");
            case 2 : printf("Are");
            case 3 : printf("You");
    }
}
```

Answer 19

```
Sorry input againHowAreYou
```

> Explanation
>
> the `default` case of the `switch-case` works when no other case value is matched to the test expression. In this question, the `default` case is evaluated first and since no other case value has been matched yet, it will execute. Since there is no `break` statement in the `default` all of the cases will be executed (**remember fallthrough?**)

Question 20

What will be the output of the following program?

```c
#include<stdio.h>
void main()
{
    int a = 15;
    if (((a >> 1) << 1) == a)
        printf("Hello");
    else
        printf("jobs");
}
```

Answer 20

```
jobs
```

Explanation

To perform the bitwise operation `>>` on variable `a` we will have to convert the decimal value `15` to binary value `1111`. **Remember**, both these values are equal, just represented differently.

After `>>`

`1111` (15) will be shifted to the right by 1 digit and will become,

`0111` (7)

After `<<`

`0111` (7) will be shifted to the left by 1 digit and will become,

`1110` (14)

Hence, after the operation, the value of `a` is not equal to the value it was in the beginning.

(((a >> 1) << 1) == a)

(((15 >> 1) << 1) == 15)

(((7) << 1) == 15)

((7 << 1) == 15)

((14 == 15)

Hence, the evaluation of this expression will result in *false*.