Department of Computer Science & Engineering Faculty of Engineering & Technology (ITER)

Assignment-2 Answers →

1) Using selection and repitition structures instead of goto has several advantages →

i) Readability → These structures make code easier to read and follow, while goto can lead to confusing spaghetti code.

ii) Structured Programming → Control structures support a clean, logical flow, while goto disrupts this flow.

iii) Debugging → Predictable flow in loops and conditionals simplifies debugging; goto makes it harder to trace issues.

iv) Avoiding Errors → Loops and conditionals are less likely to cause infinite loops or logic errors compared to goto.

v) Modularity: Control structures promote organised, reusable code, goto can create tightly coupled, difficult-to-maintain code.

2) The purpose of the break statement within a switch case is to exit the switch block once a matching case is executed. Without it, the program will continue to execute the code for all subsequent cases, which might lead to unintended behaviour.

Yes, a switch case can work without break, but this usually causes the fall-through behaviour unless intentionally desired.

Example with break:

```
int day = 2;
switch (day) {
    case 1:
        printf ("Monday");
        break;
    case 2:
        printf ("Tuesday");
        break;
    case 3:
        printf ("Wednesday");
        break;
}
```

Output → Tuesday

Example without break →

```
int day = 2;
switch (day) {
    case 1:
        printf ("Monday");
    case 2:
        printf ("Tuesday");
    case 3:
        printf ("Wednesday");
}
```

output → Tuesday Wednesday

Name: ——————————

Regd. Number: ——————————

3) Functions of break and continue statements in loops:

i) break Statement:

a) Purpose: Exits the loop immediately, regardless of the loop's condition. The program control is transferred to the first statement after the loop.

b) Application: It is used when a specific condition is met, and you need to stop further iterations of the loop.

ii) Continue statement →

a) Purpose: Skips the current iteration of the loop and moves to the next iteration without terminating the entire loop.

b) Application: It is used when you want to skip certain iterations but continue looping for others.

Examples →

break in loop →

```
for (int i = 0; i<10; i++){
  if (i==5){
    printf("Number 5 found, exiting loop.\n")
    break;
  }
  printf("i = %d\n",;i);
}
```

Output →

i = 0
i = 1
i = 2
i = 3
i = 4
Number 5 found, exiting loop.

continue in loop →

```
for (int i = 0; i < 10, i++){
    if (i % 2 == 0){
        continue;
    }
    printf("i = %d\n", i);
}
```

Output →

i = 1
i = 3
i = 5
i = 7
i = 9

Q4) When to use if-else, switch or loop →

a) if-else statements →

i) Use when → Conditions are complex and involve various comparisons or ranges.

ii) Criteria → • Multiple rational operators
• Non-discrete or complex conditions.

Eg →
```
if (age >= 18){
    printf("Minor");
}else if (age < 65){
    printf("Adult");
}else{
    printf("Senior");
}
```

ii) Switch statements →

a) Use when: Comparing a single variable against multiple ~~constant~~ discrete values (like integers or characters).

b) Criteria → • A single variable with distinct, known values.
• More readable than many if else conditions.

eg → switch (day) {

case 1:

printf ("Monday");

break;

case 2:

printf ("Tuesday");

break;

case 3:

printf ("Wednesday");

break;

}

iii) Loops (for, while, do-while) →

a) Use when → Repeating a block of code multiple lines.

b) Criteria : for → known number of iterations.

while → condition based repitition

do-while → execute at least one

eg. → for (int i=0; i<5; i++){

print ("i = %d\n", i);

}

Criteria for Selection →

i) Clarity → Use the structure that makes code mode readable.

ii) Condition Complexity → Use if-else for complex conditions, switch for discrete values.

iii) Repitition → Use loops for repeating tasks.

Q5) a) Output → returns 0

Explanation → i) y != (x - 10.0) translates to 10.0 != (25.0 - 10.0)

which is 10.0 != 15.0

ii) As 1st condition is true & x = x - 10.0, making x = 25.0 - 10.0 = 15.0

b) Output → returns 0

Explanation → i) $y < 15.0$ is true $(10.0 < 15.0)$

ii) Next, ~~y=0~~ $y > 0.0$ is also true $(10.0 >= 0.0)$

iii) Hence, $x = 5 * y = 5 * 10.0 = 50.0$

c) Explanation → While loop runs from $i = 0$ to $i = 5$, printing two numbers: i & $10 - i$.

Output →

0	10
1	9
2	8
3	7
4	6
5	5

d) Explanation → While ~~statement~~ condition is missing.

Output → Error

e) Explanation → The for loop runs with j from 1 to 5, and for each iteration, i increases by 2.

Output →

1	1
3	2
5	3
7	4
9	5

f) Explanation → i) The while (--count + 1) adds 1 to the decremented value of count in each iteration.

ii) As count is decremented in the while loop, it will continue until count reaches 0, at which point the loop stops because --count becomes -1, & the overall result of --count + 1 becomes 0 (which is false)

Output → Count down is 0

g) Explanation → i) The outer loop runs from m=9 to m>0, and the inner loop runs from n=6 to n>1.

ii) For each value of m, the inner loop prints ##### five times

iii) Total number of times it prints is 9*5 = 45

Output → ######### (45 times)

h) Explanation → i) Since there are no break statements, the switch block will "fall through".

ii) The program will print "Hello" three times because it enters the default case and continues through the remaining cases

Output → Hello Hello Hello

i) Explanation → while(i++) starts with i=0, since i++ evaluates to 0 on the first pass, the loop never runs.

Output → Nothing is printed as the loop doesn't execute.

j) Explanation → if (a=0) is an assignment, not a comparison. It sets a to 0, making the condition false, so the printf statement is never executed.

Output → Nothing is printed because the condition is false.

Q6)
```
#include <stdio.h>
int leap(int year){
    if (year % 4 == 0){
        if (year % 100 == 0){
            if (year % 400 == 0) return 1;
            else return 0;
        } else return 1;
    }
    return 0;
}
```

```
int day_of_year(int day, int month, int year){
    int days_in_month[] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
    if (leap(year)) days_in_month[1] = 29;
    int day_number = 0;
    for(int i=0; i<month-1; i++){
        day_number += days_in_month[i];
    }
    day_number += day;
    return day_number;
}
int main(){
    int day, month, year;
    printf("Enter day, month and year: ");
    scanf("%d %d %d", &day, &month, &year);
    int day_number = day_of_~~number~~year(day, month, year);
    printf("The day number for %d-%d-%d is %d\n", day, month, year,
            day_number);
    return 0;
}
```

Output->

```
Enter day, month and year: 24 12 ~~2015~~ 2016
The day number for 24-11-2016 is: 329
```

Q7)

```
#include <stdio.h>
int main(){
    int marks;
    char grade;
    printf("Enter ~~grade~~ marks obtained by student: ");
    scanf("%d", &marks);
```

```
switch (marks/10){
        case 10:
        case 9:
            grade = 'O';
            break;
        case 8:
            grade = 'A';
            break;
        case 7:
            grade = 'B';
            break;
        case 6:
            grade = 'C';
            break;
        case 5:
            grade = 'D';
            break;
        case 4:
            grade = 'E';
            break;
        case default:
            grade = 'F';
            break;
}
printf("The grade is: %c\n", grade);
return 0;
}
```

Output →

Enter marks obtained by the student: 100
The grade is: O

Q8)
```
#include <stdio.h>
#include <math.h>
double log_approximation(double x){
    double sum = 0.0;
    double term;
    double fraction = (x-1)/x;
    for(int n=1; n<=9; n++){
        term = (1.0/2.0) * pow(fraction, n);
        sum += term;
    }
    return sum;
}
int main(){
    double x, result;
    printf("Enter the value of x (x>0): ");
    scanf("%lf", &x);
    if(x<=0){
        printf("Invalid input! Please enter a value greater than 0.\n");
        return 1;
    }
    result = log_approximation(x);
    printf("Approximation of ln(%.lf) using the first 9 terms of the series
        is : %.lf\n", x, result);
    printf("Actual value of ln(%.lf) is : %lf\n", x, log(x));
    return 0;
}
```

Output →

Enter the value of x (x>0): 5
Approximation of ln(5.000000) using the first 9 terms of the series is : 1.731565
Actual value of ln(5.000000) is : 1.609438

Q9)
```
#include <stdio.h>
void printPattern(char choice){
    int i, j, k;
    for(i=0; i<= choice - 'A'; i++){
        for(j=0; j<= choice - 'A' - i; j++){
            print("%c", 'A'+j);
        }
        for(k=0; k<2*i-1; k++){
            printf(" ");
        }
        for(j= choice - 'A' - i; j>=0; j--){
            if(j != choice - 'A'){
                printf("%c", 'A'+j);
            }
            printf("\n");
        }
    }
int main(){
    char choice;
    printf("Enter the choice of the characters: ");
    scanf("%c", &choice);
    printPattern(choice);
    return 0;
}
```

Output → Enter the choice of the character: G

```
ABCDEFGFEDCBA
ABCDEF  FEDCBA
ABCDE    EDCBA
ABCD      DCBA
ABC        CBA
```

Q10)
```
#include<stdio.h>
void multiplication_table(int n){
    printf("+------------------------------------------+\n|");
    for(int i=1; i<=10; i++){
        printf("%4d", n*i);
    }
    printf("|\n|");
    for(int i=1; i<=10; i++){
        printf("%4d",i);
    }
    printf("|\n|");
    for(int i=1; i<=10;i++){
        printf("%4d",n);
    }
    printf("|\n");
    printf("+------------------------------------------+\n");
}
int main(){
    int num;
    printf("Enter the number>");
    scanf("%d", &num);
    multiplication_table(num);
    return 0;
}
```

Output → Enter the number >10

```
+------------------------------------------+
| 10  20  30  40  50  60  70  80  90 100 |
|  1   2   3   4   5   6   7   8   9  10 |
| 10  10  10  10  10  10  10  10  10  10 |
+------------------------------------------+
```