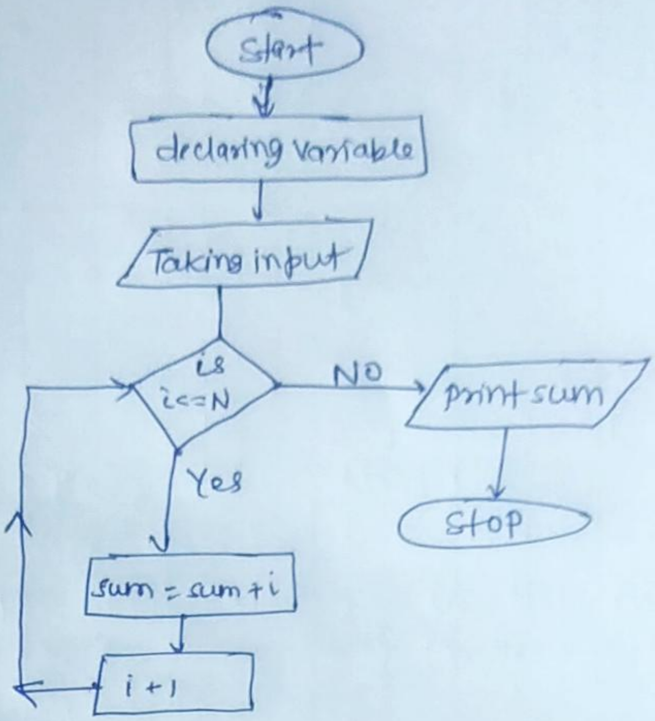
Name :- Rishikesh Kumar
Roll :- 20103051
Subject :- C programming
Exam : End sem
Branch : CSE
Date : 23-09-2021

Q.1 A compiler is a program that translates a source ~~code~~ program written in some high-level programming language into machine code.
It translates the code written in one programming language to some other language without changing the meaning of the code.
The compiler also makes the end code efficient which is optimised for execution time and memory space.
The compiling process includes basic translation mechanisms and error detection.

Q.2 Algorithm is a step-by-step procedure; which defines a set of instructions to be execute in a certain order to get the desired output. Algorithms are generally created independent of underlying languages i.e. an algorithm can be implemented in more than one programming language.
Flowchart is a diagrammatic representation of sequence of logical steps of a program. It use simple geometric shapes to depict processes and arrows to show relationship and process flow.

Q2. flowchart



Algorithm:

```
Step ①: Start
Step 2: Get value of N
Step 3: Initialize i = ϕ, s = 0;
Step 4: If (i > n) go to step 8.
Step 5: s = s + i
Step 6: i = i + 1
Step 7: go to step 3
Step 8: Display the value of s
Step 9: Stop
```

3. $(68)_{10} + (AFC)_{16} + (34)_6$

$= (68)_{10} + (10\times16^2 + 15\times16^1 + 12\times16^0)_{10} + (3\times6^1 + 4\times6^0)_{10}$

$= (68)_{10} + (2560 + 240 + 12)_{10} + (18+4)_{10}$

$= (68)_{10} + (2812)_{10} + (22)_{10} = (2902)_{10}$

converting to base 2.

2	2902	0
2	1451	1
2	725	1
2	362	0
2	181	1
2	90	0
2	45	1
2	22	0
2	11	1
2	5	1
2	2	0
	1	

$(101101010110)_2$ ans

4. A c programming, a loop is used to ~~represent~~ repeat a block of code until the specified condition is met. A loop statement allows us to execute a statement or group of statement multiple times.

~~Loops and~~ Loop type and description:

1. while loop:- It repeats a statement or group of statements while a given condition is true. It tests the condition before executing the loop body.
2. for loop: Executes a sequence of statements multiple times and abbreviated the code that manages the loop variable. Also checks the condition

before execution

(3) do...while loop: It works like a while statement, except for the part, ~~where~~ it checks the condition at the end of loop. Meaning the code will execute at least one.

(4) Nested loops:- This is not a different kind of loop. It is just a loop inside another loop. You can use any loop inside any other loop.

Q.8.
```
# include <stdio.h>
int main() {
    int n;
    printf(" Enter the number: ");
    scanf("%d", &n);

    int temp = n, rev = 0;
    while (temp) {
        rev = rev* 10 + (temp%10);
        temp /= 10;
    }

    if (rev == n) printf("%d is a palindrom", n);
    else printf ("%d is not a palindrom", n);
    return 0;
}
```

Terminal

```
Enter the number: 121
121 is a ~~pla~~ palindrom.
```

Q.9. Binary seach is a searching algorithm as the name suggests. This algorithm can be implemented on sorted array. In this algorithm we directly comes ~~at~~ the centre of array and check if the required number is ~~before~~ that centre element if yes we come out of the loop else if the element is smaller than the mid element we discards the right half and search again in left half and we do similar ~~so~~ if the element is greater than the mid element.
Binary seach is a huge optimisation over classic linear search, ~~Time complexity of binary search is O(N) while that of~~ Time complexity : Binary search ~~is~~ O(logN)
linear search O(N)

```
#include <stdio.h>
int main() {
      int arr[] = {1,2,3,4,5,6,7,8};
      int n = sizeof(arr)/sizeof(arr[0]);
      int k;
      printf(" Enter the number: ");
      scanf("%d", &k);
      int l = 0, h = n-1, ans = -1;
      while (l <= h) {
            int mid = (l+h)/2;
            if (arr[mid] > k) h = mid-1;
            else if (arr[mid] < k) l = mid+1;
            else {
                  ans = mid;
                  break;
            }
if (ans = =-1) printf ("The number is not in the array");
else printf ("The number is found at index %d", ans);
return 0;
}
```

Terminal

```
Enter the number: 5
The number is found at index 4.
```

Q.11
```
#include <stdio.h>
#include <math.h>
int main(){
        int a, b, c;
        printf("Enter the coefficients of x^2, x and constant
                term respectively: ");
        scanf("%d %d %d", &a, &b, &b);
       int D = b* b - 4* a * c;
       if(D<0) {
            printf(" The roots are imaginary\n")
            printf("The roots are (-%d +%di/2 * %d) and
            (-%d -%di/2 * %d)", b, sqrt(D), a, b, sqrt(D), a);
      } else if (D==0) {
           printf(" The roots are real and equal\n");
           printf(" The root will be %d", -b/2* a);
      }
```

```
else {
    printf(" The roots are real and unequal\n");
    printf(" The roots are %d and %d", (-b + sqrt(D))/(2*a, (-b - sqrt(D))/
                                                              (2*a);
}
 return 0;
}
```

Q.10 Bubble sort is a basic sorting algorithm. It is used to sort an array in ascending or descending order. We ~~for~~ run the loop (n-1) times and in each ~~lo~~loop we check for every consecutive element as swap the accordingly, i.e., if to be sorted ascending order then greater element will go right and vice-versa.

```
#include <stdio.h>
int main() {
        int n;
        Printf(" Enter the length of array: ");
        scanf("%d", &n);

        Int arr[n];
        printf(" Enter the elements: ");

        for (int i=0 ; i<n ; i++) scanf("%d", &arr[i]);

        for (int i=0; i <n-1 ; i++) {
                for (int j=0; j<n-i-1; j++) {
```

```
            if (arr[j] > arr[j+1]) {
                arr[j] = arr[j] ^ arr[j+1] ;
                arr[j+1] = arr[j] ^ arr[j+1] ;
                arr[j] = arr[j] ^ arr[j+1];
            }
        }
    }
    printf("The sorted array is :  ");
    for (int i = 0; i<n; i++) printf("%d", arr[i]);
    return 0;
}
```

Terminal :

```
        Enter the number of elements: 5
        Enter the elements : 2 3 5 1 6
        The sorted array is : 1 2 3 5 6
```

Q.13. (main file)

```
    # include <stdio.h>
    # include "count vowel.h"
    int main() {
        char sent [30];
        printf("Enter the sentence : ");
        scanf("%[^\n]*c", & sent );
    printf(" The number of vowel present is %d", countVowel(sent));
    return 0;
}
```

(header file)

countVowel.h

```
int ~~Co~~ countVowel( char s[30] )~~;~~ {
    int count = 0;
    for (int i=0; s[i] != '\0'; i++) {
        char c = s[i];
        if (c=='a' || c=='e' || c=='i' || c=='o' || c=='u'
           || c=='A' || c=='E' || c=='I' || c=='O' || c=='U')
        { count ++~~;~~; }
    }
    return count;
}
```

Q.15. Recursion is a ~~[illegible]~~ very popular method in which a ~~program~~ function calls ~~[illegible]~~ intself inside the function. It can make ~~[illegible]~~ very hard looking program easy to implement. There is a base case in this type of function which returns or helps the ~~bo~~ function break or else thes function could go on forever and finaly give stack overflow problem.

```
# include <stdio.h>
int factorial (int);
~~int~~
```

```
int main() {
        int n;
        printf("Enter the number: ");
        scanf("%d", &n);
        printf("The factorial of %d is %d", n, factorial(n));
        return 0;
}
int factorial(int n) {
    if(n == 0) return 1;  // Base case
    return n * factorial(n-1);  // recursive part
}
```

Terminal

```
Enter the number: 5
The factorial of 5 is 120.
```

Terminal (for Q13)

```
Enter the sentence: Prince is a good boy
The number of vowels present is 7.
```

Q.19. A switch case is an special kind of if-else statement or branching statement in C. ~~This~~ It only works for integer or character datatypes. ~~we can~~ It gives our program a better look and makes it readable. If a certain ~~condi~~ case

case is met then all the cases down below it will be executed. Sometimes this is useful sometimes it is not so to prevent this type of situation we use break statement in between the sentences.

Example:

```
        switch (c) {
            case 'a':
                printf(" This is a ");
                break;
            case 'b':
                printf(" This is b ");
                break;
            default:
                ~~[illegible]~~ printf(" This is different");
        }
```

Q.16.
```
#include <stdio.h>
typedef struct {
        char branch [30], name [30];
        int roll;
} student;

int main () {
    student arr[10];
    printf("Enter the details of students: \n");
    .
```

```
for(int i=0; i<70; i++) {
        printf("Enter the roll: ");
        ~~scanf(~~
        scanf("%d", &arr[i].roll);
        printf("Enter the Name: ");
        scanf(" %[^\n]*c", &arr[i].name);
        printf("Enter the branch: ");
        scanf(" %[^\n]*c", &arr[i].branch);
    }
    for (int i=0; i<70; i++) {
        printf("The student %s in branch %s has roll %d\n",
                  arr[i].name, arr[i].branch, arr[i].roll);
    }
    return 0;
}
```

Terminal

Output too long

* Structure : It is a composite data type declaration that defines a physically grouped list of variables under one name in a block of memory, allowing the different variables to be accessed via a single pointer or by the struct name which returns the same address.

Q.5 The differences between primary and secondary memory are:

Primary	Secondary
primary memory allocates internal memory	Secondary memory allocates ~~auxi~~ auxiliary memory
primary memory is accessed by data bus	Secondary memory is accessed by I/O channels
Directly accessed by processing unit.	not directly accessed
Primary storage devices as costlier	Secondary devices are cheaper than primary
volatile	non volatile
Ex: - RAM, ROM, etc	Ex: Hard disk etc,

Q.6 Operating system: It is the software used to operate the ~~th~~ hardware parts in computer

The main functions of operating systems are

i) Manage the computer's resource such as the central processing unit, memory, disk, drives, etc.

2) Establish user interface

3) Execute and provide services for application softwares.

4) Memory Management

5) processor Management

6) Device Management.

7) Time sharing