Q.1 Given is a C pattern program.

```
#include <stdio.h>
int main() {
  int rows = 4;
  int i, j, space;
  for (i = rows; i >= 1; --i) {
    for (space = 0; space < rows - i; ++space)
      printf("  ");
    for (j = i; j <= 2 * i - 1; ++j)
      printf("* ");
    for (j = 0; j < i - 1; ++j)
      printf("*");
    printf("\n");
  }
  return 0;
}
```

Expected Output :

```
* * * * * * *
  * * * * *
    * * *
      *
```

Received Output :

```
* * * * ***
  * * * **
    * * *
      *
```

Q.2 Given is the C programs.

```
#include<stdio.h>
int fun()
{
static int count = 2;
count = count * 2;
return count<=10?count+20:count-6;
}

int main()
{
printf("%d ", fun());
printf("%d ", fun());
printf("%d ", fun());
printf("%d ", fun());
return 0;
}
```

Expected Output :
22 24 28 10

Received Output :
24 28 10 26

Q.3 Given is a C union program.

```
#include <stdio.h>
union abc
{
   int a;
   char b;
};
int main()
{
   union abc *ptr;
   union abc var;
   var.a= 90;
   ptr = var;
   printf("The value of a is : %d", ptr->a);
   return 0;
}
```

Expected Output :
The value of a is : 90

Received Output :
Compilation Error

Q.4 Given is the program with SHAH-RUKH-KHAN as input.

```
#include <stdio.h>
#include <string.h>
void reverse_string(char*, int, int);
void reverse_string(char *x, int start, int end)
{
char ch;
if (start >= end)
return;
ch = *(x+start);
*(x+start) = *(x+end);
*(x+end) = ch;
reverse_string(x, ++start, --end);
}
int main()
{
char string_array[150];
scanf("%s", &string_array);
reverse_string(string_array, 0, strlen(string_array)+1);
printf("\nReversed String is: %s",string_array);
return 0;
}
```

Expected Output :

SHAH-RUKH-KHAN

Reversed String is: NAHK-HKUR-HAHS

Received Output :

SHAH-RUKH-KHAN

Reversed String is:

Q.5 Given is the program to check if given number is Armstrong Number or Not.

```
#include<stdio.h>
 int main()
{
int n=153;
int r,sum=0,temp;
temp=n;
while(n>0)
{
r=n%10;
sum=sum+(r*r);
n=n/10;
}
if(temp==sum)
printf("armstrong  number ");
else
printf("not armstrong number");
return 0;
}
```

Expected Output :
armstrong number

Received Output :
not armstrong number

Q.6 Given is a C program.

```
#include <stdio.h>
#include <math.h>
int main()
{  int side = 3, area, temp;
temp = sqrt(3) / 4 ;
area = temp * side * side ;
printf("Area of Equilateral Triangle is: %f",area);
return 0;
}
```

Expected Output :
Area of Equilateral Triangle is: 3.897114

Received Output :
Area of Equilateral Triangle is: 0.000000

Q.7 Given is a program to print prime numbers in given range.

```
#include <stdio.h>
int main()
{
int num1= 10, num2 = 20, flag_var, i, j;
printf("Prime numbers from %d and %d are:\n", num1, num2);
        for(i=num1+1; i<num2; ++i)
        {
                flag_var=0;
               for(j=2; j<=i/2; ++j)
               {
                        if(j%i==0)
                             {
                                    flag_var=1;
                                    break;
                             }
               }
        if(flag_var==0)
        printf("%d\n",i);
        }
        return 0;
}
```

Expected Output :

Prime numbers from 10 and 20 are:
11
13
17
19.

Received Output :

Prime numbers from 10 and 20 are:
11
12
13
14
15
16
17
18
19

Q.8 Given is a C program to find the largest element.

```
#include <stdio.h>
int main(int arg, int *args) {
  int n = 6;
  double arr[] = {50,23,19,1,32,5};
  for (int i = 1; i < n; ++i) {
    if (arr[0] < arr[i]) {
      arr[0] = arr[i];
    }
   }
  }

  printf("Largest element = %.2lf", arr[0]);
  return 0;
}
```

Expected Output :
Largest element = 50.00

Received Output :
Compile Time Error

Q.9 - 10 Given is a C program implementing a graph.

```
#include <stdio.h>
#include <stdlib.h>
#define N 6
struct Graph
{
   struct Node* head[N];
};
struct Node
{
   int dest;
   struct Node* next;
} n1;
struct Edge {
   int src, dest;
};
struct Graph* createGraph(struct Edge edges[], int n)
{
   struct Graph* graph = (struct Graph*)malloc(sizeof(struct Graph));
   for (int i = 0; i < N; i++) {
      graph->head[i] = NULL;
   }
   for (int i = 0; i < n; i++)
   {
      int src = edges[i].src;
      int dest = edges[i].dest;
      struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
      newNode->dest = src;
      newNode->next = graph->head[src];
      graph->head[src] = newNode;
   }
   return graph;
}
```

```
void printGraph(struct Graph* graph)
{
   for (int i = 0; i < N; i++)
   {
      struct Node* ptr = graph->head[i];
      while (ptr != NULL)
      {
         printf("(%d —> %d)\t", i, ptr->dest);
         ptr = ptr->next;
      }
      printf("\n");
   }
}
int main(void)
{
   struct Edge edges[] =
   {
      {0, 1}, {1, 2}, {2, 0}, {2, 1}, {3, 2}, {4, 5}, {5, 4} };
   int n = sizeof(edges[0])/sizeof(edges[0]);
   struct Graph *graph = createGraph(edges, n);
   printGraph(graph);
   return 0;
}
```

Expected Output :

(0 —> 1)
(1 —> 2)
(2 —> 1) (2 —> 0)
(3 —> 2)
(4 —> 5)
(5 —> 4)

Received Output :

(0 —> 0)

JUNIOR SECTION

Q.1 Given is the program to find first five terms of the Fibonacci Series.

```
#include <stdio.h>
void printFibonacciNumbers(int n)
{
        int f1 = 0, f2 = 1, i;
        if (n < 1)
                return;
        printf("%d ", f1);
        for (i = 1; i < n; i++) {
                printf("%d ", f2);
                int next = f1 + f2;
                f1 = f2;
                f2 = next;
        }
}
void main()
{
   unsigned long numberOfTerms = 5;
        printFibonacciNumbers(numberOfTerms);
}
```

Expected Output :

0 1 1 2 3

Received Output :

0 1 1 2 3 5

Q.2 Given is the program to check if given number is Prime or Not.

```
#include <stdio.h>
int main(int a) {
  int ironMan = 10;
  int i, flag = 0;
  if (ironMan == 0 || ironMan == 1)
    flag = 1;
  for (i = 2; i <= ironMan / 2; ++i) {
    if (ironMan % i == 0) {
      flag = 1;
      break;
    }
  }
  if (flag == 1)
    printf("%d is a prime number.", ironMan);
  else
    printf("%d is not a prime number.", ironMan);
  return -5;
}
```

Expected Output :
10 is not a prime number.

Received Output :
10 is a prime number.

Q.3 Given is the program to check if given number is a palindrome or not.

```
#include"stdio.h"
#include<maths.h>
int main()
{
   int number = 11;
   int r,sum=0,temp;
   temp=number;
   while(number>0)
      {
         r=number%10;
         sum=(sum*10)+r;
         number=number/10;
      }
   if(temp==sum)
      printf("palindrome number ");
   else
      printf("not palindrome");
   return;
}
```

Expected Output :
palindrome number

Received Output :
Compilation Error

Q.4 Given is the program implementing a structure.

```
#include <stdio.h>
struct student {
    char name[50];
    int roll;
    float marks;
} s;

int main() {
    struct student s1;
    s.roll = 20;
    s1.roll = 100;
    s1.marks = 100;
    printf("Displaying Information:\n");
    printf("Name: ");
    printf("Roll number: %d\n", s1.roll);
    printf("Marks: %.1f\n", s1.marks);
    return 0;
}
```

Expected Output :

Displaying Information:
Name: Roll number: 20
Marks: 100.0

Received Output :

Displaying Information:
Name: Roll number: 100
Marks: 100.0

Q.5 Given is the program to print a matrix.

```
#include <stdio.h>
int main()
{
  int rows = 3;
  int columns = 3;
  int k=1;
    int a[rows][columns];
    int i=1;
  while(i<=rows)
  {
     int j=1;
    do
     {   printf("%d\t",k);
         k = k + 1;
         j = j + 1;
         i = i + 1;
     }while(j<=columns);
     printf("\n");
  }
}
```

Expected Output :

1 2 3
4 5 6
7 8 9

Received Output :

1 2 3

Q.6 Given is a C program.

```
#include <stdio.h>
int main()
{
        int a = -10, b = 20;
        if(a < 0 && b > 0)
                a++;
        else if(a < 0 && b < 0)
                a--;
        else if(a < 0 && b > 0)
                b--;
        else
                b--;
        printf("%d\n",a + b);
        return 0;
}
```

Expected Output :
9

Received Output :
11

Q.7 Given is a program to find if given year is Leap year or not.

```
#include <stdio.h>
int main() {
  int year = 2011;
  if (year % 400 == 0) {
    printf("%d is a leap year.", year);
  }
  else if (year % 100 == 0) {
    printf("%d is not a leap year.", year);
  }
  else if (year % 4 == 0) {
    printf("%d is a leap year.", year);
  }
  else {
    pritf("%d is not a leap year.", year);
  }
  return 0;
}
```

Expected Output :
2011 is not a leap year.

Received Output :
Compile Time Error

Q.8 Given is a C program for memory allocation and printing the elements.

```
#include <stdio.h>
#include <stdlib.h>
#include<math.h>
#include<time.h>
#include<float.h>
int main()
{
	int* lollipop;
	int n = 10;
	int i;
	lollipop = (int*)malloc(n* 10000);
	if (lollipop == NULL) {
		printf("Memory not allocated.\n");
		exit();
	}
	else {
		printf("Memory successfully allocated using malloc.\n");
		for (i = 0; i < n; ++i) {
			lollipop[i] = i + 1;
		}
		printf("The elements of the array are: ");
		for (i = 0; i < n; ++i) {
			printf("%d, ", lollipop[i]);
		}
	}
	return 0;
}
```

Expected Output :

Memory successfully allocated using malloc.
The elements of the array are: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10,

Received Output :

Compile Time Error

Q.9 Given is a program to find factorial of a number.

```
#include<stdio.h>
long factorial(int n)
{  if (n == 0)  return 1; else  return(n + factorial(n-1));}
int main()
{
  int number = 10;
  long fact;
  fact = factorial(((number)));
  printf("Factorial of %d is %ld\n", number, fact);
  return -20152103;
}
```

Expected Output :
Factorial of 10 is 3628800

Received Output :
Factorial of 10 is 56

Q.10 Given is a program to find the Greatest number.

```
#include <stdio.h>
int main()
{
double num1, num2, num3;
num1= 10 ;
num2= 45 ;
num3= 80 ;
if (num1 >= num2 && num1 >= num3)
        printf("%lf is the largest number.", num1);
if (num2 >= num1 && num2 >= num3)
        printf("%lf is the largest number.", num2);
if (num3 >= num1 && num3 >= num2
        printf("%lf is the largest number.", num3);
return 0;
}
```

Expected Output :
80.000000 is the largest number.

Received Output :
Compile Time Error