1) C program for check whether the number is even or odd

Program:

```
#include <stdio.h>
 int main()
{
  int num;

 Printf("Enter the number ");
 scanf ("%d", &num);

 if (num %2 ==0)
 {                 %d                     num
   Printf("/n      is a even number",);
 }
else
 {                %d                       num
   Printf("/n      is a odd number");
 }
return 0;
}
```

Sample input :-

5

output :-

5 is a odd number

2. Sum of first n numbers

INPUT :- 5

OUTPUT :- 15

3. Even Sum given 1 to n range using while

INPUT :- 6

OUTPUT :- 12

4. Reverse a number

INPUT :- 12345

OUTPUT :- 54321

5. Palindrome or not

INPUT :- 12321

OUTPUT :- It is a palindrome

6. Check whethere Armstrong number or not

INPUT :- 153

OUTPUT :- It is a armstrong number

7. Factorial without recursion

INPUT :- 5

OUTPUT :- 120

8. Factorial with recursion

INPUT :- 3

OUTPUT :- 6

9. Fibanocci without recursion

INPUT :- 5

OUTPUT :- 0 1 1 2 3

10. Fibanocci with recursion

INPUT :- 5

OUTPUT :- 0 1 1 2 3

11. Search an element in array using linear Search

INPUT :- 5 (Size)

6 7 8 9 10 (elements)

8 (position to be search)

OUTPUT

8 found at index 2

12. Search an element in array using binary search

INPUT :- 6 (Size)

8 9 7 6 5 4 (elements)

5 (position to be search)

OUTPUT :-

5 found at index 4.

3. Sum of elements in array

INPUT :- 5 (Size)

1 2 3 4 5 (element)

OUTPUT :- 15 (SUM)

14. Merge array

INPUT :- 5 (Size 1)

1 2 3 4 5 (elem 1)

6 (Size 2)

1 2 3 4 5 6 (elem2)

OUTPUT :- 1 2 3 4 5 1 2 3 4 5 6

15. INSERTION & DELETION at middle

INPUT :- 5 (Size)

1 2 3 4 5 (element

3 (element to be deleted)

OUTPUT :- 1 2 4 5.

16. STRING REVERSE

INPUT :- BHANU

OUTPUT :- UNAHB

17. STRING PALINDROME

INPUT :- EYE

OUTPUT :- It is a pallindrome

18. Element search in string

INPUT :- BHANU (string name)

A (element to search)

OUTPUT :- 3

19. No. of vowels in string

INPUT :- BHANU

OUTPUT :- 2

20 Matrix MULTIPLICATION

INPUT :- 3 (ROWS
3 (Columns)

1 2 3 4 5 6 7 8 9 (1st Matrix)

1 2 3 4 5 6 7 8 9 (2nd Matrix)

OUTPUT :-

```
30    36    42

66    81    96

102   126   150
```

21. STRING MANIPULATION

INPUT :- BHANU (Str1)
TEJA (Str2)

(CAT) BHANUTEJA
(COPY) BHANUTEJA
(LENGTH) 9
~~£0~~ (CMP) Not equal

25. INFIX to POSTFIX expression usingstack

INPUT :- a+b*c/(d-e).

OUTPUT :- abc*de-/+

23. STACK IMPLEMENTATION

INPUT :- SELECT OPTION(1 PUSH, 2 POP, 3 SHOW, 4 END)

OUTPUT :- (1) PUSH 1
(1) PUSH 2
(1) PUSH 3
(1) PUSH 4
(2) POP 4
(3) SHOW ELEMENTS IN STACK
1 2 3
(4) END

24. Queue IMPLEMENTATATION

INPUT :- SELECT OPTION (1 Eneque, 2 Dequeue, 3 size, 4 Front, 5 Rear, 6 show, 7 exit)

(1) 1
(1) 2
(1) 3
(1) 4
(1) 5
(1) 6
} enequeue

(2) Dequeue (1)
(3) ~~2~~ size< 5
(4) 2 (Front)

(5) 6 (Rear)
(6) 2 3 4 5 6
(7) stops

26. Evaluating expression using Stack

INPUT : 524*+1-

OUTPUT : 12

36. Minimum Spanning tree using Prim's algorithm

INPUT :-

6 (No. of nodes)

adjacency Matrix

```
0  3  1  6  0  0
3  0  5  0  3  0
1  5  0  5  6  4
6  0  5  0  0  2
0  3  6  0  0  6
0  0  4  2  6  0
```

OUTPUT :-

~~Edge~~ (1,3) cost : 1
(1,2) cost : 3
(2,5) cost : 3
(3,6) cost : 4
(6,4) cost : 2
Minimum cost = 13

35. (Minimum) Shortest path using Dijistra's

INPUT :- 5 (Vertices)

```
 0  10   0  30  100 | 30   0  20   0  60
10   0  50   0    0 | 100  0  10  60   0
 0  50   0  20   10 |
```

Starting node : 0

OUTPUT:

D of node 1 = 10

P :=1<-0

D: of node 2 = 50

Path = 2<-3<-0

D. of node3 =30

Path =3<-0

D. of node4 =60

path =4<-2<-3<-0

37. Minimum spanning tree using Kruskal's algorithm

INPUT :- No. of vertices : 6

cost adjacency matrix :

```
0 3 1 6 0 0
3 0 5 0 3 0
1 5 0 5 6 4
6 0 5 0 0 2
0 3 6 0 0 6
0 0 4 2 6 0
```

OUTPUT :-

```
1 edge (1,3) =1
2 edge (4,6)=2
3 edge (1,2)=3
4 edge (2,5)=3
5 edge (3,6)=4
Minimum cost =13
```

38. DFS

INPUT :- 8

```
0 1 1 0 0 1 0 0
1 0 0 1 0 0 0 0
1 1 0 0 0 0 0 1
1 1 1 0 0 0 0 1
1 1 1 0 0 1 0 0
1 0 0 0 0 0 0 1
1 1 0 1 0 0 0 0
1 1 1 0 0 0 0 1
```

OUTPUT

```
0
1
5
2
7
6
3
4
```

39 BFS

INPUT :- 4

```
1 3 4 2
2 3 4 5
1 2 5 7
2 1 6 4
```

Starting vertex 1

OUTPUT :

```
1 2 3 4
```