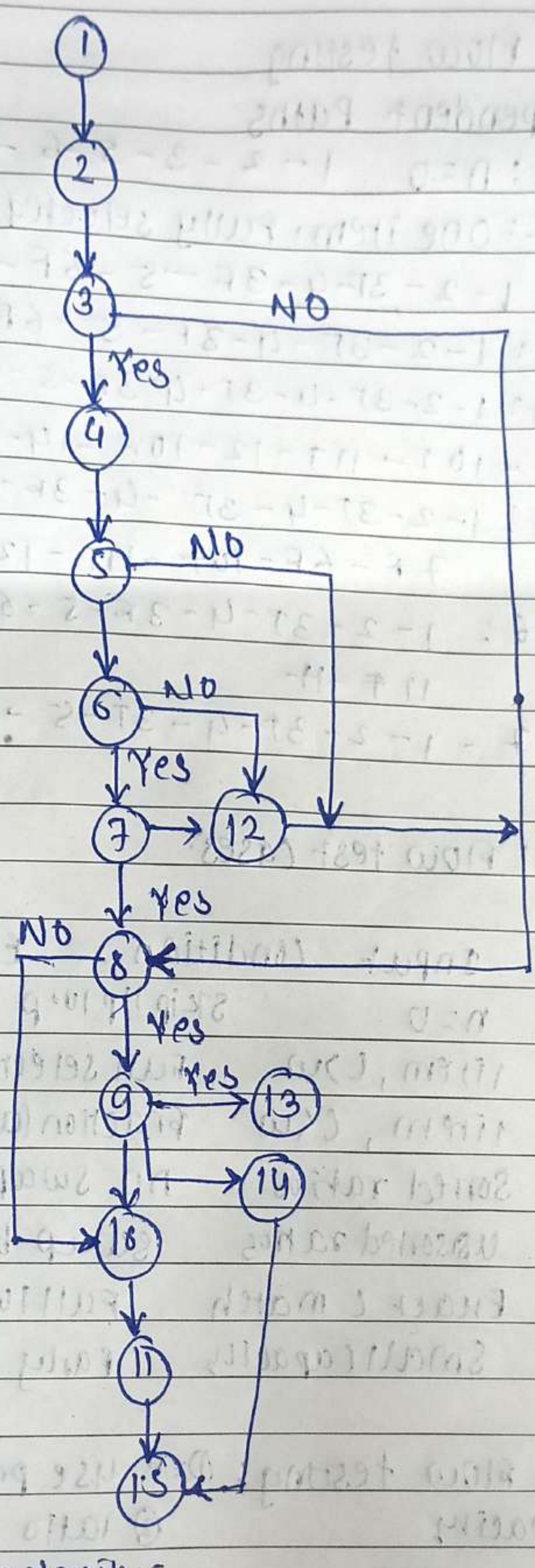
Assignment No.3

- Write a test cases using white box testing for the given c code.
  1. Calculate cyclomatic complexity
  2. Control Flow testing
  3. Data flow testing

→ C Program for fractional knapsack

```
#include <stdio.h>
struct Item {
        int profit;
        int weight;
        float ratio;
  };
void swap (struct Item *a, struct Item *b) {
   s     struct Item temp = *a;
          *a = *b;
          *b = temp:
}
int main() {
      int n, i, j;
      float capacity, totalprofit = 0.0;
      printf("enter no. of items: ");
      scanf("%d", &n);
      struct Item items[n];
      for(i=0; i<n; i++) {
          printf("Enter profit and weight for
                      item %d: ", i+1);
          scanf("%d %d", &items[i].profit, &items[i].weight);
          items[i].ratio = (float) items[i].profit /
                                items[i].weight; }
 } printf("Enter knapsack capacity: ");
  scanf("%f", &capacity);
```

```
for(i=0 ; i< n-1; i++) {
      for (j=0 ;j < n-j-1 ; j++){
         if(items[j].ratio <items[j+1].ratio){
             swap(&items[j], &items[j+1]);
         }
      }
}
for (i=0: i<n ; i++) {
     if ((capacity >= items[i].weight){
          capacity ~~x~~ -= items[i].weight;
          totalprofit += items[i]. profit;
     } else{
          totalprofit += items[i].ratio*(capacity;
          break;
     }
}
printf("Maximum profit = %.2f\n", totalprofit);
return 0;
}
```



cyclomattc complexity=

no.of nodes(N)=15

no.of Edges(E)=20

$V(G) = E-N+2 = 20-15+2$

$V(G) = 7$

$\boxed{\text{cyclomattc complexity} = 7}$

Control flow testing

- Independent paths

Path 1: n=0 1-2-3-5-6-10-14-15

Path 2: one item, fully selected

Path 1-2-3T-4-3F-5-6F-10T-11T-12-10F-1

Path 3: 1-2-3T-4-3F-5-6F-10T-11F-13-14-15

Path 4: 1-2-3T-4-3T-4-3F-5-6T-7T-8F-7F-6F-
-10T-11T-12-10F-14-15

Path 5: 1-2-3T-4-3T-4-3F-5-6T-7T-8T-9-
7F-6F-10T-11T-12-10F-14-15

Path 6: 1-2-3T-4-3F-5-6F-10T-11T-12
- 11F-14-

Path 7 - 1-2-3T-4-3T-5-6F-11F-13

Control flow test cases

TC	Input	Condition	Expected result
1	n=0	skip i/p loop	Profit =0
2	1 item, C>w	Full selection	Profit =full
3	1 item, C<w	fraction case	partial profit
4	Sorted ratios	no swap	correct order
5	unsorted ratios	swap branch	correct sorted
6	Exact C match	Full loop exe	exact profit
7	Small capacity	Early loop	fraction result

Data flow testing: Def-use pairs

① capacity

Def - scanf

use - if (C>=w)

use - cap -=w

② ratio

Def - profit/weight

use - if (ratio comparison)

③ total profit

Def - total profit =0

use - totalprofit += profit

use - totalprofit += ratio*C

- ① Data Flow test cases

TC	Input	coverage
1	n=0	minimal def-use
2	single item full	c & p def-use
3	single item fraction	r & c def use
4	multiple items sorted	no swap def-use
5	multiple items unsorted	swap def-use
6	mixed case	All def use pa