THE UNIVERSITY OF DODOMA

COLLEGE OF INFORMATICS AND VIRTUAL EDUCATION

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING (CSE)

INDIVIDUAL ASSIGNMENTS

DATA STRUCTURES AND ALGORITHMS

CP 213

COURSE : BSc. SOFTWARE ENGINEERING.

REGISTRATION NUMBER: T22-03-09187.

1. Store keeper is required to record stock of item in inventing Management System. Each item has item name(string), item id(character), item quantity(integer). From Data structure point of view write a program that will accept record of 20 items use:
    i. Structure concept.

Code to implement

StokeKeeper.cpp > main()

```cpp
#include<iostream>
using namespace std;
struct Items{
    string name;
    char Id[20];
    int Quantity;

}Item[20];
int main(){
    for(int i=0;i<20;i++){
        cout<<"Enter Item Name: \n";
        cin>>Item[i].name;
        cout<<"Enter Item Id: \n";
        cin>>Item[i].Id;
        cout<<"Enter Item Quantity: \n";
        cin>>Item[i].Quantity;

    }
    cout<<"The Item information is: \n";
    for(int j=0;j<20;j++){
        cout<<"Item Number"<<j+1<<endl;
        cout<<"Item Name: "<<Item[j].name<<endl;
        cout<<"Item id: "<<Item[j].Id<<endl;
        cout<<"Item Quantity: "<<Item[j].Quantity<<endl;
        cout<<endl;
    }
}
```

Output on Running.

```
mastesa@mastesa-HP-EliteBook-840-G3:~/Documents/project/Assignment$ g++ -o keeper1 StokeKeeper.cpp
mastesa@mastesa-HP-EliteBook-840-G3:~/Documents/project/Assignment$ ./keeper1
Enter Item Name:
water
Enter Item Id:
1
Enter Item Quantity:
20
Enter Item Name:
Soda
Enter Item Id:
2
Enter Item Quantity:
30
Enter Item Name:
vinegar
Enter Item Id:
3
Enter Item Quantity:
50
Enter Item Name:
Mayonise
Enter Item Id:
4
Enter Item Quantity:
12
Enter Item Name:
Bites
Enter Item Id:
5
Enter Item Quantity:
30
Enter Item Name:
Yorghurt
Enter Item Id:
6
Enter Item Quantity:
50
Enter Item Name:
Fresh Milk
Enter Item Id:
```

```
Enter Item Quantity:
50
Enter Item Name:
Cakes
Enter Item Id:
10
Enter Item Quantity:
30
Enter Item Name:
Pencils
Enter Item Id:
11
Enter Item Quantity:
56
Enter Item Name:
Rubber
Enter Item Id:
12
Enter Item Quantity:
30
Enter Item Name:
Pens
Enter Item Id:
13
Enter Item Quantity:
20
Enter Item Name:
Bread
Enter Item Id:
14
Enter Item Quantity:
40
Enter Item Name:
Sugar
Enter Item Id:
15
Enter Item Quantity:
40
Enter Item Name:
Salt
Enter Item Id:
```

```
The Item information is:
Item Number1
Item Name: FreshMilk
Item id: 1
Item Quantity: 20

Item Number2
Item Name: Water
Item id: 2
Item Quantity: 20

Item Number3
Item Name: Soda
Item id: 3
Item Quantity: 30

Item Number4
Item Name: Bites
Item id: 4
Item Quantity: 40

Item Number5
Item Name: Soap
Item id: 5
Item Quantity: 30

Item Number6
Item Name: Detergents
Item id: 6
Item Quantity: 40

Item Number7
Item Name: Bisquits
Item id: 7
Item Quantity: 40

Item Number8
Item Name: Sweets
Item id: 8
Item Quantity: 30
```

```
Item Number9
Item Name: Chocolate
Item id: 9
Item Quantity: 50

Item Number10
Item Name: Cakes
Item id: 10
Item Quantity: 30

Item Number11
Item Name: Pencils
Item id: 11
Item Quantity: 56

Item Number12
Item Name: Rubber
Item id: 12
Item Quantity: 30

Item Number13
Item Name: Pens
Item id: 13
Item Quantity: 20

Item Number14
Item Name: Bread
Item id: 14
Item Quantity: 40

Item Number15
Item Name: Sugar
Item id: 15
Item Quantity: 40

Item Number16
Item Name: Salt
Item id: 16
Item Quantity: 70

Item Number17
```

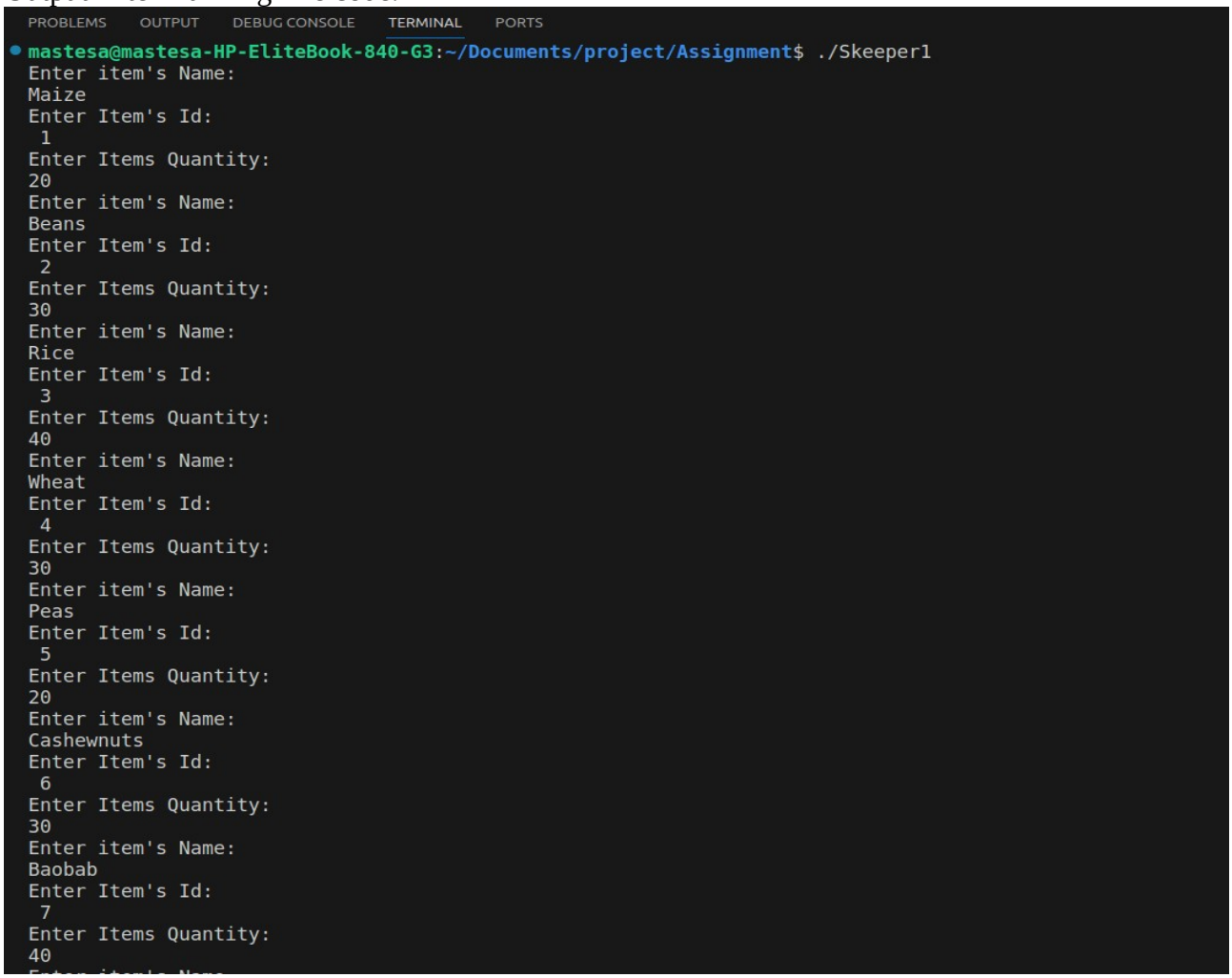
```
Item Number17
Item Name: Vannila
Item id: 17
Item Quantity: 100

Item Number18
Item Name: Perfumes
Item id: 18
Item Quantity: 20

Item Number19
Item Name: CookingOil
Item id: 19
Item Quantity: 60

Item Number20
Item Name: Kerosene
Item id: 20
Item Quantity: 60

mastesa@mastesa-HP-EliteBook-840-G3:~/Documents/project/Assignment$
```

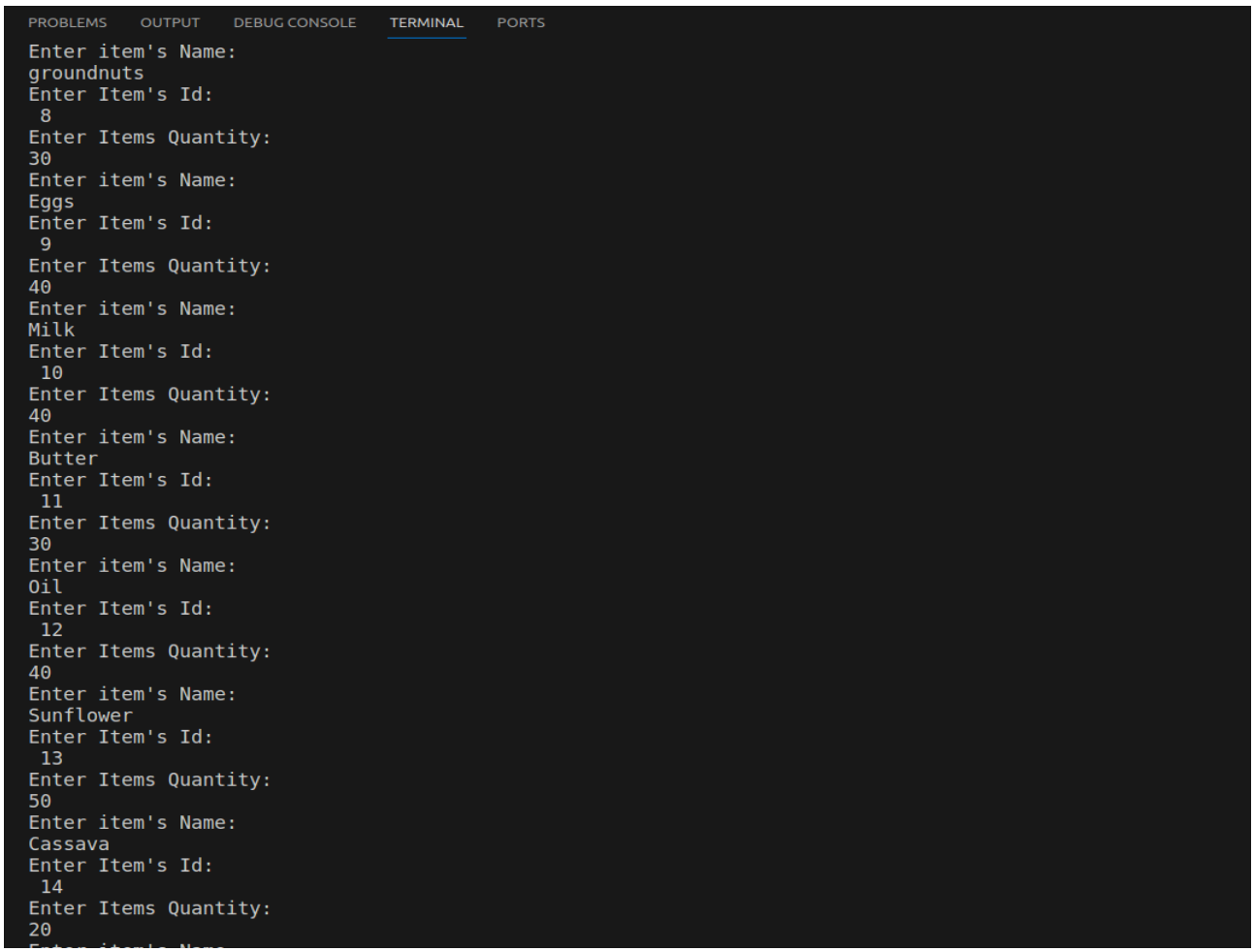
ii. Linked List concept.

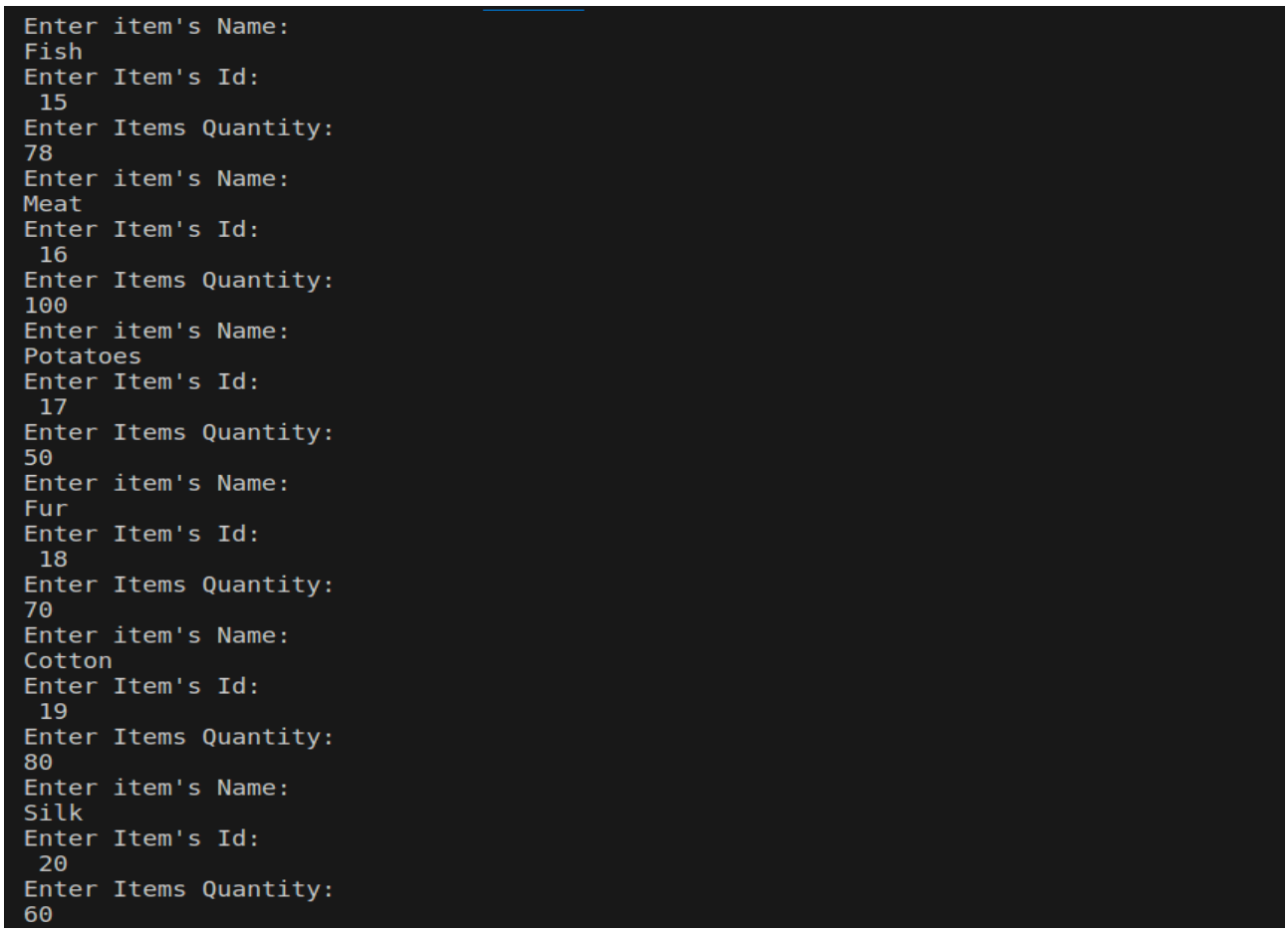
LinkedS_keeper.cpp > Items > next

```cpp
#include<iostream>
using namespace std;
struct Items{
    string name;
    char Id[20];
    int Quantity;
    Items *next;
};
int main(){
    Items *head=NULL;
    for (int i=0;i<20;i++){
        Items *item =new Items();
        cout<<"Enter item's Name: \n";
        cin>>item->name;
        cout<<"Enter Item's Id: \n ";
        cin>>item->Id;
        cout<<"Enter Items Quantity: \n";
        cin>>item->Quantity;
        item->next=NULL;
            if(head==NULL){
                head=item;
            }
            else{
                Items *px=head;
                while(px->next!=NULL){
                    px=px->next;
                }
                px->next=item;
            }
    }
    int counter=1;
    Items *copy=head;
    while(copy!=NULL&& counter<=20){
        cout<<"Item Number "<<counter<<"; \n";
        cout<<"Name: "<<copy->name<< endl;
        cout<<"Id: "<<copy->Id<<endl;
        cout<<"Quantity: "<<copy->Quantity<<endl;
        cout<<endl;
        counter++;
        copy=copy->next;
    }
}
```

Output After Running The code.




PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
Enter item's Name:
groundnuts
Enter Item's Id:
8
Enter Items Quantity:
30
Enter item's Name:
Eggs
Enter Item's Id:
9
Enter Items Quantity:
40
Enter item's Name:
Milk
Enter Item's Id:
10
Enter Items Quantity:
40
Enter item's Name:
Butter
Enter Item's Id:
11
Enter Items Quantity:
30
Enter item's Name:
Oil
Enter Item's Id:
12
Enter Items Quantity:
40
Enter item's Name:
Sunflower
Enter Item's Id:
13
Enter Items Quantity:
50
Enter item's Name:
Cassava
Enter Item's Id:
14
Enter Items Quantity:
20


Enter item's Name:
Fish
Enter Item's Id:
15
Enter Items Quantity:
78
Enter item's Name:
Meat
Enter Item's Id:
16
Enter Items Quantity:
100
Enter item's Name:
Potatoes
Enter Item's Id:
17
Enter Items Quantity:
50
Enter item's Name:
Fur
Enter Item's Id:
18
Enter Items Quantity:
70
Enter item's Name:
Cotton
Enter Item's Id:
19
Enter Items Quantity:
80
Enter item's Name:
Silk
Enter Item's Id:
20
Enter Items Quantity:
60

```
Item Number 1;
Name: Maize
Id: 1
Quantity: 20

Item Number 2;
Name: Beans
Id: 2
Quantity: 30

Item Number 3;
Name: Rice
Id: 3
Quantity: 40

Item Number 4;
Name: Wheat
Id: 4
Quantity: 30

Item Number 5;
Name: Peas
Id: 5
Quantity: 20

Item Number 6;
Name: Cashewnuts
Id: 6
Quantity: 30

Item Number 7;
Name: Baobab
Id: 7
Quantity: 40

Item Number 8;
Name: groundnuts
Id: 8
Quantity: 30

Item Number 9;
Name: Eggs
Id: 9
```

```
Id: 12
Quantity: 40

Item Number 13;
Name: Sunflower
Id: 13
Quantity: 50

Item Number 14;
Name: Cassava
Id: 14
Quantity: 20

Item Number 15;
Name: Fish
Id: 15
Quantity: 78

Item Number 16;
Name: Meat
Id: 16
Quantity: 100

Item Number 17;
Name: Potatoes
Id: 17
Quantity: 50

Item Number 18;
Name: Fur
Id: 18
Quantity: 70

Item Number 19;
Name: Cotton
Id: 19
Quantity: 80

Item Number 20;
Name: Silk
Id: 20
Quantity: 60

mastesa@mastesa-HP-EliteBook-840-G3:~/Documents/project/Assignment$
```

2. With reference to the code in 1(B) write a piece of code that will :
    i. Add one item after 10th item.

Codes:

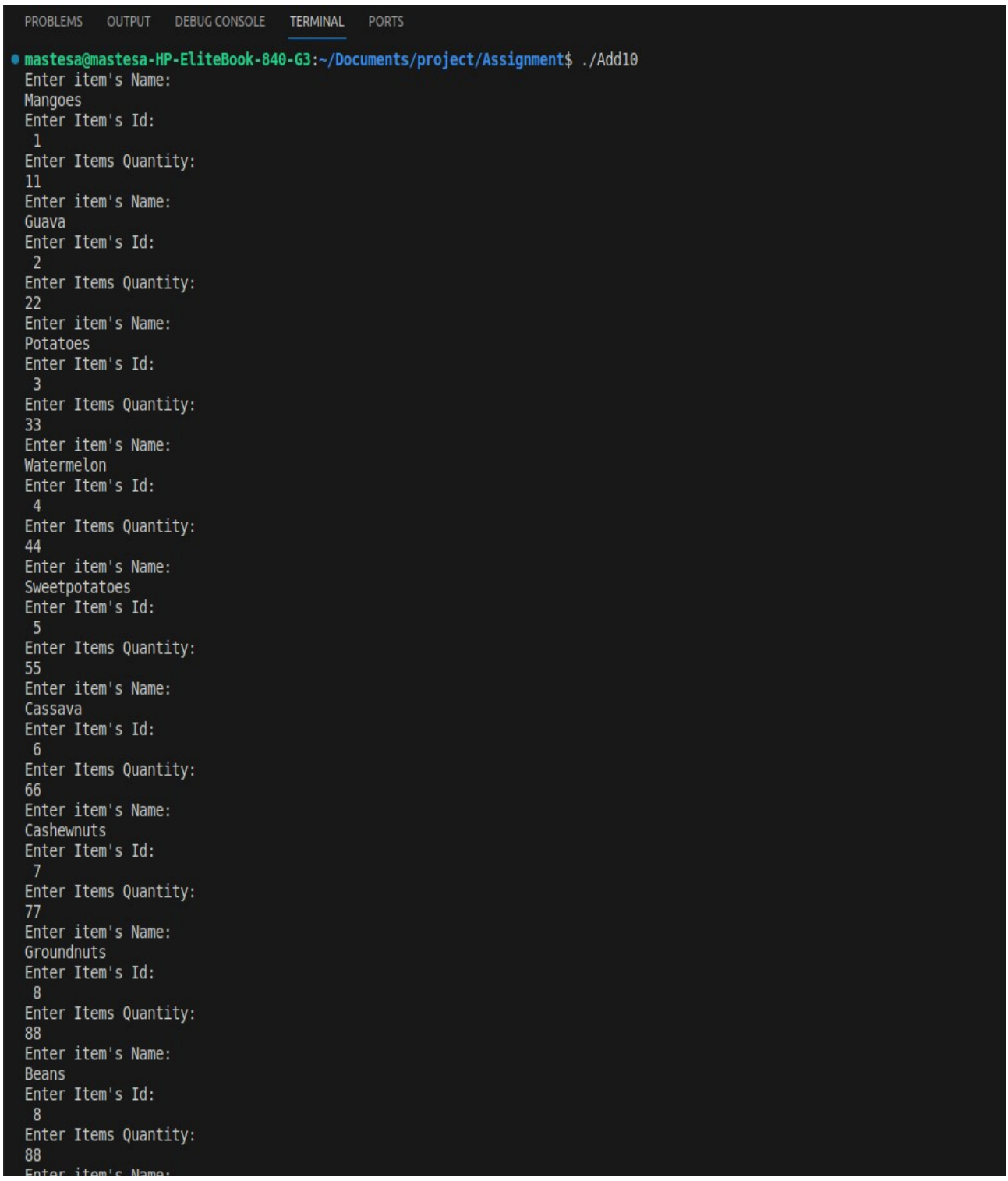
Addat10.cpp > main()

```cpp
#include<iostream>
using namespace std;
struct Items{
    string name;
    char Id[20];
    int Quantity;
    Items *next;

};
int main()
{
    Items *head=NULL;
    for (int i=0;i<;i++){
        Items *item =new Items();
        cout<<"Enter item's Name: \n";
        cin>>item->name;
        cout<<"Enter Item's Id: \n ";
        cin>>item->Id;
        cout<<"Enter Items Quantity: \n";
        cin>>item->Quantity;
        item->next=NULL;
            if(head==NULL){
                head=item;
            }
            else{
                Items *px=head;
                while(px->next!=NULL){
                    px=px->next;
                }
                px->next=item;
            }
    }
```

Addat10.cpp > main()

```
    }
    int counter=1;
    Items *copy=head;
    while(copy!=NULL&& counter<=20){
        cout<<"Item Number "<<counter<<"; \n";
        cout<<"Name: "<<copy->name<< endl;
        cout<<"Id: "<<copy->Id<<endl;
        cout<<"Quantity: "<<copy->Quantity<<endl;
        cout<<endl;
        counter++;
        copy=copy->next;
    }
    //ADD array After node10
        Items *item =new Items();
        cout<<"Enter item's Name: \n";
        cin>>item->name;
        cout<<"Enter Item's Id: \n ";
        cin>>item->Id;
        cout<<"Enter Items Quantity: \n";
        cin>>item->Quantity;
        item->next=NULL;
    Items *ptr=head;
    for(int j=0;j<9;j++){
        ptr=ptr->next;
    }
    item->next=ptr->next;
    ptr->next=item;
    counter=1;
    copy=head;
    while(copy!=NULL&& counter<=21){
        cout<<"Item Number "<<counter<<"; \n";
        cout<<"Name: "<<copy->name<< endl;
        cout<<"Id: "<<copy->Id<<endl;
        cout<<"Quantity: "<<copy->Quantity<<endl;
        cout<<endl;
        counter++;
        copy=copy->next;

}
```

Output of the code:


PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
mastesa@mastesa-HP-EliteBook-840-G3:~/Documents/project/Assignment$ ./Add10
Enter item's Name:
Mangoes
Enter Item's Id:
1
Enter Items Quantity:
11
Enter item's Name:
Guava
Enter Item's Id:
2
Enter Items Quantity:
22
Enter item's Name:
Potatoes
Enter Item's Id:
3
Enter Items Quantity:
33
Enter item's Name:
Watermelon
Enter Item's Id:
4
Enter Items Quantity:
44
Enter item's Name:
Sweetpotatoes
Enter Item's Id:
5
Enter Items Quantity:
55
Enter item's Name:
Cassava
Enter Item's Id:
6
Enter Items Quantity:
66
Enter item's Name:
Cashewnuts
Enter Item's Id:
7
Enter Items Quantity:
77
Enter item's Name:
Groundnuts
Enter Item's Id:
8
Enter Items Quantity:
88
Enter item's Name:
Beans
Enter Item's Id:
8
Enter Items Quantity:
88

```
Enter item's Name:
Maize
Enter Item's Id:
 9
Enter Items Quantity:
99
Enter item's Name:
Coffee
Enter Item's Id:
 10
Enter Items Quantity:
100
Enter item's Name:
tea
Enter Item's Id:
 11
Enter Items Quantity:
110
Enter item's Name:
Onions
Enter Item's Id:
 12
Enter Items Quantity:
120
Enter item's Name:
Carrots
Enter Item's Id:
 13
Enter Items Quantity:
88
Enter item's Name:
Tomatoes
Enter Item's Id:
 14
Enter Items Quantity:
140
Enter item's Name:
carrots
Enter Item's Id:
 15
Enter Items Quantity:
39
Enter item's Name:
Peas
Enter Item's Id:
 16
Enter Items Quantity:
30
Enter item's Name:
Lotion
Enter Item's Id:
 17
Enter Items Quantity:
88
Enter item's Name:
Water
Enter Item's Id:
```

```
Item Number 1;
Name: Mangoes
Id: 1
Quantity: 11

Item Number 2;
Name: Guava
Id: 2
Quantity: 22

Item Number 3;
Name: Potatoes
Id: 3
Quantity: 33

Item Number 4;
Name: Watermelon
Id: 4
Quantity: 44

Item Number 5;
Name: Sweetpotatoes
Id: 5
Quantity: 55

Item Number 6;
Name: Cassava
Id: 6
Quantity: 66

Item Number 7;
Name: Cashewnuts
Id: 7
Quantity: 77

Item Number 8;
Name: Groundnuts
Id: 8
Quantity: 88

Item Number 9;
Name: Beans
Id: 8
Quantity: 88

Item Number 10;
Name: Maize
Id: 9
Quantity: 99

Item Number 11;
Name: Coffee
Id: 10
Quantity: 100

Item Number 12;
Name: tea
Id: 11
```

```
Id: 13
Quantity: 88

Item Number 15;
Name: Tomatoes
Id: 14
Quantity: 140

Item Number 16;
Name: carrots
Id: 15
Quantity: 39

Item Number 17;
Name: Peas
Id: 16
Quantity: 30

Item Number 18;
Name: Lotion
Id: 17
Quantity: 88

Item Number 19;
Name: Water
Id: 18
Quantity: 400

Item Number 20;
Name: Juice
Id: 19
Quantity: 990
```

Declaring a new node with item name called “WHITEFLOUR”

```
Quantity: 990

Enter item's Name:
WHITEFLOUR
Enter Item's Id:
 21
Enter Items Quantity:
55
```

Result of new node being added after node 10

```
Item Number 10;
Name: Maize
Id: 9
Quantity: 99

Item Number 11;
Name: WHITEFLOUR
Id: 21
Quantity: 55

Item Number 12;
Name: Coffee
Id: 10
Quantity: 100
```

ii. Delete the 7^{th} item from stack

Lines of code:

Delete7.cpp > main()

```cpp
#include<iostream>
using namespace std;
struct Items{
    string name;
    char Id[20];
    int Quantity;
    Items *next;

};
int main(){
    Items *head=NULL;
    for (int i=0;i<20;i++){
        Items *item =new Items();
        cout<<"Enter item's Name: \n";
        cin>>item->name;
        cout<<"Enter Item's Id: \n ";
        cin>>item->Id;
        cout<<"Enter Items Quantity: \n";
        cin>>item->Quantity;
        item->next=NULL;
            if(head==NULL){
                head=item;
            }
            else{
                Items *px=head;
                while(px->next!=NULL){
                    px=px->next;
                }
                px->next=item;
            }
    }
    int counter=1;
    Items *copy=head;
    while(copy!=NULL&& counter<=20){
        cout<<"Item Number "<<counter<<"; \n";
        cout<<"Name: "<<copy->name<< endl;
        cout<<"Id: "<<copy->Id<<endl;
        cout<<"Quantity: "<<copy->Quantity<<endl;
        cout<<endl;
        counter++;
        copy=copy->next;
    }
```

```
    //Deleting the seventh node;
    cout<<"Enter the node t delete"<<endl;
    int ptr;
    cin>>ptr;
    ptr--;
    Items *a,*b;
    a=head;
    b=a->next;
    for(int j=1;j<ptr;j++){
        a=b;
        b=b->next;
    }
    a->next=b->next;
    b->next=NULL;
    delete b;
    cout<<endl;

    counter=1;
    copy=head;
    while(copy!=NULL&& counter<=20){
        cout<<"Item Number "<<counter<<"; \n";
        cout<<"Name: "<<copy->name<< endl;
        cout<<"Id: "<<copy->Id<<endl;
        cout<<"Quantity: "<<copy->Quantity<<endl;
        cout<<endl;
        counter++;
        copy=copy->next;

}


}
```

```
mastesa@mastesa-HP-EliteBook-840-G3:~/Documents/project/Assignment$ g++ -o dele7 Delete7.cpp
mastesa@mastesa-HP-EliteBook-840-G3:~/Documents/project/Assignment$ ./dele7
Enter item's Name:
Maji
Enter Item's Id:
 1
Enter Items Quantity:
20
Enter item's Name:
Juisi
Enter Item's Id:
 2
Enter Items Quantity:
30
Enter item's Name:
Mangoes
Enter Item's Id:
 3
Enter Items Quantity:
30
Enter item's Name:
Banana
Enter Item's Id:
 4
Enter Items Quantity:
30
Enter item's Name:
Apples
Enter Item's Id:
 5
Enter Items Quantity:
40
Enter item's Name:
Guava
Enter Item's Id:
 6
Enter Items Quantity:
80
Enter item's Name:
Watermelon
Enter Item's Id:
 7
Enter Items Quantity:
50
Enter item's Name:
Pineapples
Enter Item's Id:
 8
Enter Items Quantity:
80
Enter item's Name:
Beans
Enter Item's Id:
 9
Enter Items Quantity:
34
```

```
Enter item's Name:
Maize
Enter Item's Id:
 10
Enter Items Quantity:
100
Enter item's Name:
Peas
Enter Item's Id:
 11
Enter Items Quantity:
20
Enter item's Name:
Ground nuts
Enter Item's Id:
 Enter Items Quantity:
12
Enter item's Name:
20
Enter Item's Id:
 Cashenuts
Enter Items Quantity:
13
Enter item's Name:
130
Enter Item's Id:
 Wheat
Enter Items Quantity:
14
Enter item's Name:
234
Enter Item's Id:
 Cassava
Enter Items Quantity:
15
Enter item's Name:
Potatoes
Enter Item's Id:
 16
Enter Items Quantity:
90
Enter item's Name:
Vegetables
Enter Item's Id:
 17
Enter Items Quantity:
45
Enter item's Name:
carrots
Enter Item's Id:
 18
Enter Items Quantity:
46
Enter item's Name:
Onions
Enter Item's Id:
```

```
Id: 1
Quantity: 20

Item Number 2;
Name: Juisi
Id: 2
Quantity: 30

Item Number 3;
Name: Mangoes
Id: 3
Quantity: 30

Item Number 4;
Name: Banana
Id: 4
Quantity: 30

Item Number 5;
Name: Apples
Id: 5
Quantity: 40

Item Number 6;
Name: Guava
Id: 6
Quantity: 80

Item Number 7;
Name: Watermelon
Id: 7
Quantity: 50

Item Number 8;
Name: Pineapples
Id: 8
Quantity: 80

Item Number 9;
Name: Beans
Id: 9
Quantity: 34

Item Number 10;
Name: Maize
Id: 10
Quantity: 100

Item Number 11;
Name: Peas
Id: 11
Quantity: 20

Item Number 12;
Name: Ground
Id: nuts
```

```
Item Number 10;
Name: Maize
Id: 10
Quantity: 100

Item Number 11;
Name: Peas
Id: 11
Quantity: 20

Item Number 12;
Name: Ground
Id: nuts
Quantity: 12

Item Number 13;
Name: 20
Id: Cashenuts
Quantity: 13

Item Number 14;
Name: 130
Id: Wheat
Quantity: 14

Item Number 15;
Name: 234
Id: Cassava
Quantity: 15

Item Number 16;
Name: Potatoes
Id: 16
Quantity: 90

Item Number 17;
Name: Vegetables
Id: 17
Quantity: 45

Item Number 18;
Name: carrots
Id: 18
Quantity: 46

Item Number 19;
Name: Onions
Id: 19
Quantity: 200

Item Number 20;
Name: Pawpaw
Id: 20
Quantity: 99
```

```
Quantity: 99

Enter the node t delete
7

Item Number 1:
```

```
Item Number 6;
Name: Guava
Id: 6
Quantity: 80

Item Number 7;
Name: Pineapples
Id: 8
Quantity: 80

Item Number 8;
Name: Beans
Id: 9
Quantity: 34
```

3. If the code written in 1(B) is a:
   (a) Queue then write a piece of code that:
      i. Add one item in the Queue.

AddQueue.cpp > main()

```cpp
#include<iostream>
using namespace std;
struct Items{
    string name;
    char Id[20];
    int Quantity;
    Items *next;

};
int main(){
    Items *head=NULL;
    for (int i=0;i<5;i++){
        Items *item =new Items();
        cout<<"Enter item's Name: \n";
        cin>>item->name;
        cout<<"Enter Item's Id: \n ";
        cin>>item->Id;
        cout<<"Enter Items Quantity: \n";
        cin>>item->Quantity;
        item->next=NULL;
            if(head==NULL){
                head=item;
            }
            else{
                Items *px=head;
                while(px->next!=NULL){
                    px=px->next;
                }
                px->next=item;
            }
    }
    int counter=1;
    Items *copy=head;
    while(copy!=NULL&& counter<=5){
        cout<<"Item Number "<<counter<<"; \n";
        cout<<"Name: "<<copy->name<< endl;
        cout<<"Id: "<<copy->Id<<endl;
        cout<<"Quantity: "<<copy->Quantity<<endl;
        cout<<endl;
        counter++;
        copy=copy->next;
    }
```

```
//Adding a new node in a Queue
        Items *item =new Items();
        cout<<"Enter item's Name: \n";
        cin>>item->name;
        cout<<"Enter Item's Id: \n ";
        cin>>item->Id;
        cout<<"Enter Items Quantity: \n";
        cin>>item->Quantity;
        item->next=NULL;
        //add node to Queue
        Items *trav=head;
         while(trav->next!=NULL){
            trav=trav->next;
         }
         trav->next=item;

        cout<<"New Updated List:::::::::::::::::: /n";
        counter=1;
        copy=head;
        while(copy!=NULL&& counter<=6){
            cout<<"Item Number "<<counter<<"; \n";
            cout<<"Name: "<<copy->name<< endl;
            cout<<"Id: "<<copy->Id<<endl;
            cout<<"Quantity: "<<copy->Quantity<<endl;
            cout<<endl;
            counter++;
            copy=copy->next;
        }


}
```

```
mastesa@mastesa-HP-EliteBook-840-G3:~/Documents/project/Assignment$ g++ -o add AddQueue.cpp
mastesa@mastesa-HP-EliteBook-840-G3:~/Documents/project/Assignment$ ./add
Enter item's Name:
Water
Enter Item's Id:
 1
Enter Items Quantity:
34
Enter item's Name:
Juice
Enter Item's Id:
 2
Enter Items Quantity:
40
Enter item's Name:
Sugar
Enter Item's Id:
 3
Enter Items Quantity:
45
Enter item's Name:
Soda
Enter Item's Id:
 4
Enter Items Quantity:
50
Enter item's Name:
Salt
Enter Item's Id:
 5
Enter Items Quantity:
54
Item Number 1;
Name: Water
Id: 1
Quantity: 34

Item Number 2;
Name: Juice
Id: 2
Quantity: 40

Item Number 3;
Name: Sugar
Id: 3
Quantity: 45

Item Number 4;
Name: Soda
Id: 4
Quantity: 50

Item Number 5;
Name: Salt
Id: 5
Quantity: 54
```

```
Enter item's Name:
Mangoes
Enter Item's Id:
 6
Enter Items Quantity:
100
New Updated List:::::::::::::::::: /nItem Number 1;
Name: Water
Id: 1
Quantity: 34

Item Number 2;
Name: Juice
Id: 2
Quantity: 40

Item Number 3;
Name: Sugar
Id: 3
Quantity: 45

Item Number 4;
Name: Soda
Id: 4
Quantity: 50

Item Number 5;
Name: Salt
Id: 5
Quantity: 54

Item Number 6;
Name: Mangoes
Id: 6
Quantity: 100
```

ii. Delete one item from the Queue.

DeleteQueue.cpp > main()

```cpp
#include<iostream>
using namespace std;
struct Items{
    string name;
    char Id[20];
    int Quantity;
    Items *next;

};
int main(){
    Items *head=NULL;
    for (int i=0;i<2;i++){
        Items *item =new Items();
        cout<<"Enter item's Name: \n";
        cin>>item->name;
        cout<<"Enter Item's Id: \n ";
        cin>>item->Id;
        cout<<"Enter Items Quantity: \n";
        cin>>item->Quantity;
        item->next=NULL;
            if(head==NULL){
                head=item;
            }
            else{
                Items *px=head;
                while(px->next!=NULL){
                    px=px->next;
                }
                px->next=item;
            }
    }
    int counter=1;
    Items *copy=head;
    while(copy!=NULL&& counter<=20){
        cout<<"Item Number "<<counter<<"; \n";
        cout<<"Name: "<<copy->name<< endl;
        cout<<"Id: "<<copy->Id<<endl;
        cout<<"Quantity: "<<copy->Quantity<<endl;
        cout<<endl;
        counter++;
        copy=copy->next;
    }
    //Deleting a node in a Queue
```

```
    }
//Deleting a node in a Queue
        Items *t=head;
        head=head->next;
        delete t;

        cout<<"New Updated List:::::::::::::::::: /n";
        counter=1;
        copy=head;
        while(copy!=NULL&& counter<=4){
            cout<<"Item Number "<<counter<<"; \n";
            cout<<"Name: "<<copy->name<< endl;
            cout<<"Id: "<<copy->Id<<endl;
            cout<<"Quantity: "<<copy->Quantity<<endl;
            cout<<endl;
            counter++;
            copy=copy->next;
        }


}
```

```
mastesa@mastesa-HP-EliteBook-840-G3:~/Documents/project/Assignment$ g++ -o delete DeleteQueue.cpp
mastesa@mastesa-HP-EliteBook-840-G3:~/Documents/project/Assignment$ ./delete
Enter item's Name:
water
Enter Item's Id:
 1
Enter Items Quantity:
20
Enter item's Name:
Juice
Enter Item's Id:
 2
Enter Items Quantity:
30
Enter item's Name:
Milk
Enter Item's Id:
 3
Enter Items Quantity:
40
Enter item's Name:
Soda
Enter Item's Id:
 4
Enter Items Quantity:
48
Enter item's Name:
Wine
Enter Item's Id:
 5
Enter Items Quantity:
65
Item Number 1;
Name: water
Id: 1
Quantity: 20

Item Number 2;
Name: Juice
Id: 2
Quantity: 30

Item Number 3;
Name: Milk
Id: 3
Quantity: 40

Item Number 4;
Name: Soda
Id: 4
Quantity: 48

Item Number 5;
Name: Wine
Id: 5
```

```
New Updated List:::::::::::::::::: /nItem Number 1;
Name: Juice
Id: 2
Quantity: 30

Item Number 2;
Name: Milk
Id: 3
Quantity: 40

Item Number 3;
Name: Soda
Id: 4
Quantity: 48

Item Number 4;
Name: Wine
Id: 5
Quantity: 65

mastesa@mastesa-HP-EliteBook-840-G3:~/Documents/project/Assignment$
```

(b) Stack then write a piece of code that:

i. Adds one item in Stack.

ADDStack.cpp > main()

```cpp
#include<iostream>
using namespace std;
struct Items{
    string name;
    char Id[20];
    int Quantity;
    Items *next;

};
int main(){
    Items *head=NULL;
    for (int i=0;i<5;i++){
        Items *item =new Items();
        cout<<"Enter item's Name: \n";
        cin>>item->name;
        cout<<"Enter Item's Id: \n ";
        cin>>item->Id;
        cout<<"Enter Items Quantity: \n";
        cin>>item->Quantity;
        item->next=NULL;
            if(head==NULL){
                head=item;
            }
            else{
                Items *px=head;
                while(px->next!=NULL){
                    px=px->next;
                }
                px->next=item;
            }
    }
    int counter=1;
    Items *copy=head;
    while(copy!=NULL&& counter<=5){
        cout<<"Item Number "<<counter<<"; \n";
        cout<<"Name: "<<copy->name<< endl;
        cout<<"Id: "<<copy->Id<<endl;
        cout<<"Quantity: "<<copy->Quantity<<endl;
        cout<<endl;
        counter++;
        copy=copy->next;
    }
```

```
    //ADD ITEM TO STACK
        Items *item =new Items();
        cout<<"Enter item's Name: \n";
        cin>>item->name;
        cout<<"Enter Item's Id: \n ";
        cin>>item->Id;
        cout<<"Enter Items Quantity: \n";
        cin>>item->Quantity;
        item->next=NULL;
        //Add node at the end of te list
         Items *trav=head;
         while(trav->next!=NULL){
            trav=trav->next;
         }
         trav->next=item;

        counter=1;
        copy=head;
        while(copy!=NULL&& counter<=6){
            cout<<"Item Number "<<counter<<"; \n";
            cout<<"Name: "<<copy->name<< endl;
            cout<<"Id: "<<copy->Id<<endl;
            cout<<"Quantity: "<<copy->Quantity<<endl;
            cout<<endl;
            counter++;
            copy=copy->next;
        }
}
```

```
mastesa@mastesa-HP-EliteBook-840-G3:~/Documents/project/Assignment$ g++ -o push ADDStack.cpp
mastesa@mastesa-HP-EliteBook-840-G3:~/Documents/project/Assignment$ ./push
Enter item's Name:
Water
Enter Item's Id:
 1
Enter Items Quantity:
20
Enter item's Name:
Milk
Enter Item's Id:
 3
Enter Items Quantity:
40
Enter item's Name:
Juice
Enter Item's Id:
 4
Enter Items Quantity:
50
Enter item's Name:
Wine
Enter Item's Id:
 6
Enter Items Quantity:
70
Enter item's Name:
Soda
Enter Item's Id:
 2
Enter Items Quantity:
50
Item Number 1;
Name: Water
Id: 1
Quantity: 20

Item Number 2;
Name: Milk
Id: 3
Quantity: 40

Item Number 3;
Name: Juice
Id: 4
Quantity: 50

Item Number 4;
Name: Wine
Id: 6
Quantity: 70

Item Number 5;
Name: Soda
Id: 2
Quantity: 50
```

```
Quantity: 50

Enter item's Name:
MANGOES
Enter Item's Id:
 8
Enter Items Quantity:
67
```

ii. Deletes one item from Stack.

DeleteStack.cpp > main()

```cpp
#include<iostream>
using namespace std;
struct Items{
    string name;
    char Id[20];
    int Quantity;
    Items *next;

};
int main(){
    Items *head=NULL;
    for (int i=0;i<4;i++){
        Items *item =new Items();
        cout<<"Enter item's Name: \n";
        cin>>item->name;
        cout<<"Enter Item's Id: \n ";
        cin>>item->Id;
        cout<<"Enter Items Quantity: \n";
        cin>>item->Quantity;
        item->next=NULL;
            if(head==NULL){
                head=item;
            }
            else{
                Items *px=head;
                while(px->next!=NULL){
                    px=px->next;
                }
                px->next=item;
            }
    }
    int counter=1;
    Items *copy=head;
    while(copy!=NULL&& counter<=4){
        cout<<"Item Number "<<counter<<"; \n";
        cout<<"Name: "<<copy->name<< endl;
        cout<<"Id: "<<copy->Id<<endl;
        cout<<"Quantity: "<<copy->Quantity<<endl;
        cout<<endl;
        counter++;
        copy=copy->next;
    }
```

```
43      //DElete item from the Stack
44          Items *trav=head;
45          while(trav->next->next!=NULL){
46              trav=trav->next;
47          }
48          trav->next=NULL;
49          trav=NULL;
50          counter=1;
51          copy=head;
52          while(copy!=NULL&& counter<=4){
53          cout<<"Item Number "<<counter<<"; \n";
54          cout<<"Name: "<<copy->name<< endl;
55          cout<<"Id: "<<copy->Id<<endl;
56          cout<<"Quantity: "<<copy->Quantity<<endl;
57          cout<<endl;
58          counter++;
59          copy=copy->next;
60
61          }
62  }
```

```
mastesa@mastesa-HP-EliteBook-840-G3:~/Documents/project/Assignment$ g++ -o pop DeleteStack.cpp
mastesa@mastesa-HP-EliteBook-840-G3:~/Documents/project/Assignment$ ./pop
Enter item's Name:
Milk
Enter Item's Id:
 1
Enter Items Quantity:
20
Enter item's Name:
Juice
Enter Item's Id:
 2
Enter Items Quantity:
30
Enter item's Name:
Mango
Enter Item's Id:
 3
Enter Items Quantity:
50
Enter item's Name:
Wine
Enter Item's Id:
 4
Enter Items Quantity:
40
```

```
Item Number 1;
Name: Milk
Id: 1
Quantity: 20

Item Number 2;
Name: Juice
Id: 2
Quantity: 30

Item Number 3;
Name: Mango
Id: 3
Quantity: 50

Item Number 4;
Name: Wine
Id: 4
Quantity: 40

Item Number 1;
Name: Milk
Id: 1
Quantity: 20

Item Number 2;
Name: Juice
Id: 2
Quantity: 30

Item Number 3;
Name: Mango
Id: 3
Quantity: 50
```

4. Briefly Explain the following concepts in data Structures
    i. Graph.
    A graph is a non linear data structure with no specific structure at all . A graph exists as the collection of two sets (Set of vertices and set of Edges). In graph the vertices are also termed Nodes while the edges are the arc connecting the nodes together.

    ii. General tree.
    General trees are non-linear data structures in which a node can be followed by one or more nodes it consists of a root node as the starting point of the tree and the leaves as the terminal parts of the tree and its may consist varying number of paths branching from a common node point or from the root .

    iii. Binary tree.
    A binary tree is a tree in which the root and its internal nodes are limited to having at-most two paths per node point thus each parent node can have only two or one or zero children and not otherwise thus the name binary tree.

    iv. Breadth First search and Depth First Search.
    Under Breadth first search the algorithm starts with the root of the tree then moves left to right across the upper level, then moves left to right in the lower level and so on till the lowest level. The search will go on until the required node is found or until all nodes have been examined,its not preferred for large tree search.
    Under depth first search one branch is examined thoroughly down the levels till the leaves of the branch before shifting to the next branch and so on until all branches and nodes have been examined. Its easier when we suspect a nodes possible target location hence being faster than breadth first search.

5. Write a piece of code that implements Bubble sort Algorithm. Use array age []={100, 3, 1, 7, 80} to demonstrate how your codes work.

6. Briefly explain the concept of infix , Prefix and Postfix .Show how an expression
    12/(4+2)+8-4/2*4
    is converted to postfix Expression.

Solution:

Expression: **12/(4+2)+8-4/2*4**

push operator / into stack

12

push the open parenthesis

12

push operator “+” into stack

12 4 2

The closed parenthesis will pop out all operators until it meets with the open parenthesis. So operator “+” will be poped out.

12 4 2 +

push operator “+” into the stack
This action will cause the operator “/” to be popped from the stack since it is of low precedence than “+”.

12 4 2 + /

push operator “-” into the stack while popping out “+” operator

12 4 2+ / 8+ 4

push the operator “/” into stack while popping “-”

12 4 2 +/ 8+4-2

push the operator “*” into stack

12 4 2 +/ 8+4-2 /4

pop all out

12 4 2 +/ 8+4 2 /4 * -

o

i. Infix: Is a mathematical operation category in which the operator is placed right between the operands. Its common approach in most mathematical equations
   Example: 2+3
ii. Prefix: Is a mathematical operation in which the operator is placed in front of the the operands (written before the operands)in a mathematical expression or equation.
   Example:+ 2 3 or *9 4.

iii. Postfix: Is a mathematical operation in which the operator is placed behind the operators (written after the operands being written) in a mathematical operation.
Example: 2 3+ and 9 4*.

BUBBLE SORT ALGORITHMS

1. What is the best time complexity of bubble sort? Explain.

Bubble sort – is the sorting algorithm that arranges (sorts) elements of an array and comparing each element with the element next to it (Following).

The best time complexity of bubble sort is O(n) .This occurs when the array is already sorted. Means that when there is no sorting required.

2. Assume that we use Bubble Sort to sort n distinct elements in ascending order. When the best case of Bubble Sort does occur? Explain.

The best case of bubble sort occurs when the input array is already sorted and no element swap is needed. Therefore in this case, the situation occurs when the elements are sorted in ascending order. The bubble sort will make only one path through the Array to confirm that it is sorted and time complexity of 0(n) will resulting where n is the number of elements in Array. This is because there will no swaps during the pass

3. What is the number of swapping needed to sort the numbers 8, 22, 7, 9, 31, 5, 13, in ascending order, using bubble sort? Show the answers.

1st Pass: 8 , 7 , 22 , 9 , 31 , 5 , 13

8 , 7 , 9 , 22 , 31 , 5 , 13

8 , 7 , 9 , 22 , 5 , 31 , 13

8 , 7 , 9 , 22 , 5 , 13 , 31 Swaps: 4

2nd Pass : 7 , 8 , 9 , 22 , 5 , 13 , 31

7 , 8 , 9 , 5 , 22 , 13 , 31

7 , 8 , 9 , 5 , 13 , 22 , 31 Swaps: 3

3rd Pass : 7 , 8 , 5 , 9 , 13 , 22 , 31 Swap: 2

4th Pass : 7 , 5 , 8 , 9 , 13 , 22 , 31 Swap: 1

5th Pass : 5 , 7 , 8 , 9 , 13 , 22 , 31 Swap: 1

Since the array is sorted after 5th pass , no further swaps possible , hence the total number of swapping needed to sort the numbers 8, 22, 7, 9, 31, 5, 13 is

= 4 + 3 + 1 + 1 + 1 = 10 Swaps

4. Suppose there are **n** elements in the array. What is the maximum number of comparisons that can take place when a bubble sort algorithm is implemented? Show how to deduce the total number of comparisons.

Given n elements in the array such that;

At pass 1**:**	Number of comparisons = Number of swaps = n - 1
At pass 2**:**	Number of comparisons = Number of swaps = n - 2
At pass 3**:**	Number of comparisons = Number of swaps = n - 3
At pass n -2 **:**	Number of comparisons = Number of swaps = 2
At pass n - 1**:**	Number of comparisons = Number of swaps = 1

Now, the maximum number of comparisons required to sort the array is given by ,

$N = (n - 1) + (n - 2) + (n - 3) + \ldots + 3 + 2 + 1$

$N = S_K = (K*(K + 1))/2$

$N = S_{n-1} = (n - 1)*(n - 1 + 1)/2$

$N = (n * (n - 1))/2$

5.What is the worst-case time complexity of the Bubble Sort algorithm? Explain.

The worst case time complexity of the bubble sort algorithm is, $O(n^2)$. This means that the time taken by the algorithm to sort an array grows quadratically with the size of the input

such that as the number of inputs increases, the time spent by the algorithm in sorting an array increases proportionally. The case occurs when the array is reversely sort , implying that the array is in descending order and ascending order is needed or vice versa .

6.The Bubble Sort algorithm is an example of a comparison - based sorting algorithm .Discuss.

The babble sort algorithm is an example of a comparison based on sorting algorithm because it has basing on ascending and descending order arrangements. Bubble sort is indeed a comparison based algorithm which means it sorts elements by comparing them pairwise and swapping them if they are in the wrong order.

7. Bubble sort algorithm is an example of an in - place algorithm. Discuss

Bubble sort algorithm is an example of an in place algorithm because it using swap algorithm in order to arrange the item or number into ascending and descending order also it use the programming language to implement it example C++ language.

8.In the Bubble sort algorithm, what is the result of a pass? Explain.

The result of pass it determined numerical number of swapping during sorting the numbers or item also the result of a pass is that the largest element among the unsorted elements bubbles up or shift to the last position of the current unsorted portion of the array. Also bubble sort indeed makes multiple passes through a list .It compares adjacent item and exchanges that are out of order. Each pass through the list places the next largest value in it's proper place each item bubbles up to the location where it belongs. For example the algorithm make in first pass to compare value of the first and second element in the array, check if array[0] is greater than array[1]

9.Discuss the drawbacks of the Bubble Sort Algorithm.

a. Inefficiency: bubble sort has a worst case and average case time complexity of $o(n^2)$ making it inefficient for large data sets. It involves multiple passes through the array swapping adjacent element until the entire array is sorted. The same number of comparisons and swaps, regardless of whether the array is already partially sorted or not

b. Lack of adaptiveness: bubble sort does not adapt to the input data. It always performs average case-time complexity of whether the array is already partially sorted or not.