1NH20CS125

by Manikanta P

Submission date: 12-Jan-2022 04:43PM (UTC+0530)
Submission ID: 1740546384
File name: Manikanta_P_1NH20CS125_report.pdf (974.61K)
Word count: 3319
Character count: 16899

NEW HORIZON
COLLEGE OF ENGINEERING

Autonomous College Permanently Affiliated to VTU, Approved by AICTE & UGC
Accredited by NAAC with 'A' Grade, Accredited by NBA

A MINI PROJECT REPORT

for

Mini Project in C (19CSE39)

On

THE INVENTORY MANAGEMENT SYSTEM

Submitted by

MANIKANTA P

USN - 1NH20CS125, Sem - 3, Sec - C

In partial fulfilment for the award of the degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

NEW HORIZON
COLLEGE OF ENGINEERING
Autonomous College Permanently Affiliated to VTU, Approved by AICTE & UGC
Accredited by NAAC with 'A' Grade, Accredited by NBA

CERTIFICATE

This is to certify that the mini project worktitled

DATA STRUCTURE IMPLEMENTATION INC

submitted in partial fulfilment of the degree of Bachelor of Engineering in Computer Science and Engineering by

MANIKANTA P

USN:1NH20CS125

DURING

ODD SEMESTER 2021-2022

for

Course: Mini Project in C -19CSE39

Signature of Reviewer

Signature of HOD

SEMESTER END EXAMINATION

Name of the Examiner	*Signature with date*
1. _____	_____
2. _____	_____

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be, but impossible without the mention of the people who made it possible, whose constant guidance and encouragement crowned my efforts with success.

I thank the management, **Dr. Mohan Manghnani**, Chairman of NEW HORIZON EDUCATIONAL INSTITUTIONS for providing necessary infrastructure and creating good environment.

I also record here the constant encouragement and facilities extended to me by

Dr. Manjunatha, Principal, NHCE, **Dr. Prashanth.C.S.R,** Dean Academics, **Dr. B. Rajalakshmi**, Head of the Department of Computer Science and Engineering. I extend my sincere gratitude to them.

I express my gratitude to **Ms. Bangari Sindhuja** my project reviewer for constantly monitoring the development of the project and setting up precise deadlines. Her valuable suggestions were the motivating factors in completing the work.

Finally, a note of thanks to all the teaching and non-teaching staff of Computer Science and Engineering Department for their cooperation extended to me and my friends, who helped me directly or indirectly in the course of the project work.

(MANIKANTA P)

(1NH20CS125)

ABSTRACT

INVENTORY MANAGMENT SYSTEM:

The Inventory Management system is a management system which contains adding products, deleting products, viewing products and modifying the products. The software system can store the data of products /goods of a factory/company. Inventory Management System would be able to maintain information and able to keep records of that particular event. This project can be implemented in any industries/companies by fulfilling basic requirements. This Inventory Management System will provide the information about incoming and outgoing status of goods. Writing all the (products/goods) details in a file and finding it whenever it is needed in future, is very difficult and very much time consuming. A lot of space is required to keep the files for a long time. Keeping the files for long time can damage it during the floods or any other hazardous situation.

TABLE OF CONTENTS

CHAPTERS

Chapter 1

Introduction

- The whole program is based on Inventory Management and making the Factory work easier. Which also helps the user to easily add and remove products/goods.
- The target is to make the user work easier to access the factory problems. By this Inventory Management System is one project which can deal with all the problems.
- The Inventory Management system is an environment where all the process of the users in the factory is managed. It is done through the automated computerized method. Conventionally this system is done using papers, files, and binders.
- As the number of the product increases in the factory manually managing the count becomes a hectic job for the administrator. This computerized system stores all the data in the database which makes it easy to fetch and update whenever needed.
- It includes processes like add the products, delete if you want to and also you can modify the product information and display the final inventory.

CHAPTER 2

ANALYSIS

2.1 OBJECTIVE OF THE PROJECT:

- Adding products
- Deleting products
- Display products
- Modify products

2.2 Requirement Specification

Software Requirements:

- Any C COMPILER (TURBO C++, VS CODE, DEV C++)

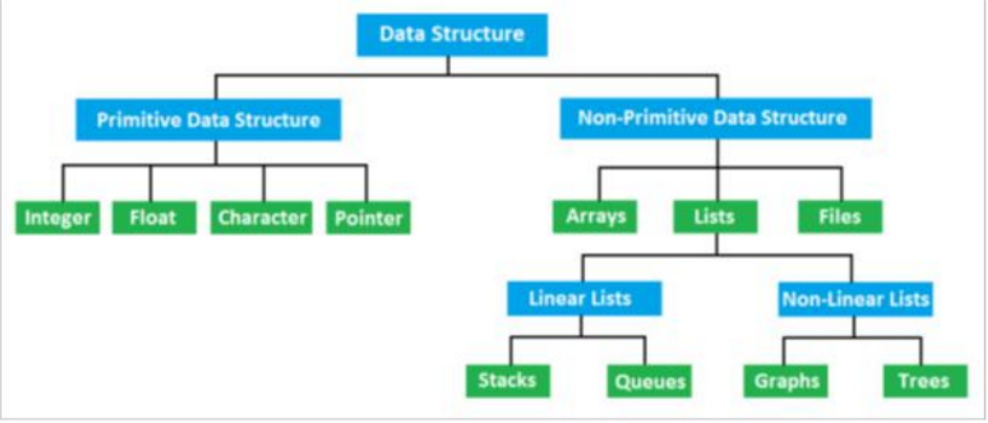
Hardware Requirements:

- Hard Disk – 2 GB+
- RAM Required – 2 GB+

CHAPTER 3

IMPLEMANTATION

3.1 DATA STRUCTURE



Data is a collection of elements and structure is the organization of these elements.
Data structure is the study of
How effectively data is collected and stored.
How effectively data is organized.
How effectively data is retrieved.
How effectively logical relations are found.

Data structures are of different structures being primitive data structures and non-primitive data structures:
Primitive data structures are data structures which can be directly manipulated using the machine instructions.
Some examples of **primitive data structures** are:
. Int
. Float
. Char
. Pointer

Non-primitive data structures are the data structures which cannot be manipulated directly using the machine instructions.
Some exampled of non-primitive data structures are:
. Arrays
. Lists
. Files
Lists are further classified into linear and non-linear lists.
Linear lists consist of:
. Stack
. Queue
. Linked list
Non-linear lists consist of:

. Trees
. Graph

The data structure implemented in this program is linked list:

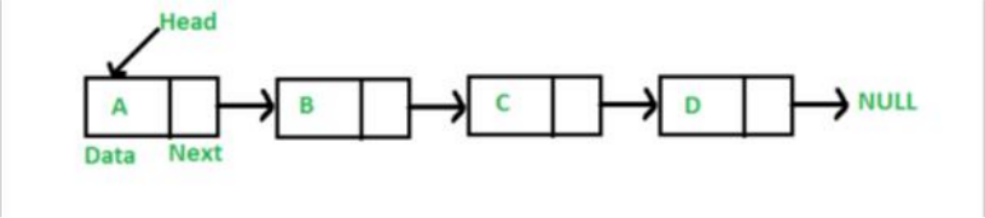


Fig.3.2.2.

Linked List

. A linked list is a linear data structure which consists of different data elements called nodes.
. Each node will have 2 parts. The first part consists of data or some value while the second part holds the address of the next node.
. Memory is allocated to the nodes using dynamic memory allocation functions such **malloc(), calloc(), realloc() and free().**

1. **malloc() :** This function is used to allocate a complete single block of memory of the specified size. It returns the starting address of the block of memory allocated and hence a pointer is used to store the address returned by malloc.
Syntax –
datatype *ptr = (datatype *)malloc(size)

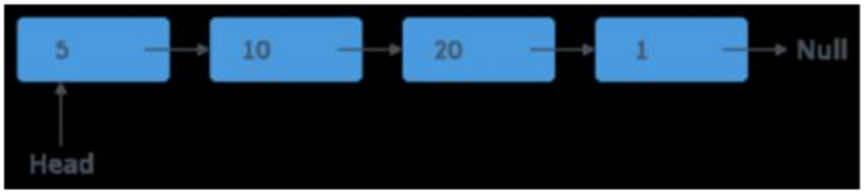
2**. calloc() :** This function allocates the specified size of memory in multiple block having the same size. Each block is assigned to NULL. It returns the starting address of the block of memory allocated and hence a pointer is used to store the address returned.
Syntax -
datatype *ptr = (datatype *)calloc(size , number of blocks)

3. **realloc() :** This function is used to reallocate the allocated memory. It returns the starting address of the new block of memory allocated and hence a pointer is used to store the address returned
Syntax -
datatype *ptr = (datatype *)realloc(ptr , size)

ptr is the name of the pointer storing the address of the memory block.
4**. free() :** This function is used to free the allocated memory. The parameter passed in free() is the pointer storing the address of the allocated memory block.

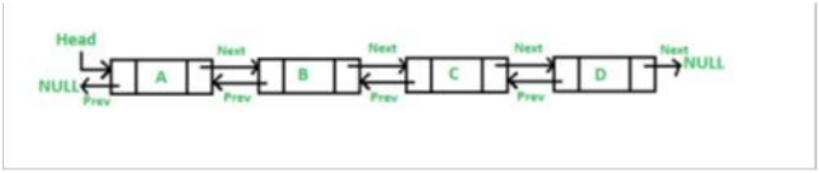
Syntax – free(pointer name)
. There are 4 different types of linked lists:

1.Single Linked List (SLL)



A single linked list is a linked list which consists of a group of nodes where each node has 2 parts. One part is the data stored in the linked list and the other is the address of the next node. The address part of the last node always stores the value NULL. A pointer (Head in above figure) points to the first node of the linked list and is used to access the linked list.

2.Double Linked List (DLL):



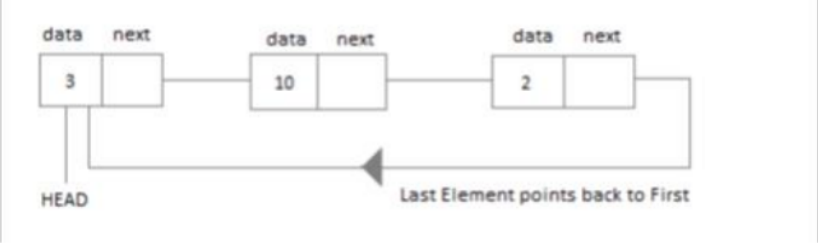
A double linked list is a linked list which consists of a group of nodes where each node has 3 parts. One part is the data stored in the linked list, the second part is the address of the next node and the third part is the address of the previous node. The next address part of the last node points to NULL and similarly the previous address part of the first node points to NULL.
Doubly linked list, despite of occupying more memory than singly linked lists, are more commonly used as you can traverse from the first node to the last and vice versa, making it easier to access the nodes.

Circular Linked List (CLL)

A circular linked is a list in which the address part of the last node stores the address of the first node. Circular linked lists can be single circular linked lists or double circular linked lists.

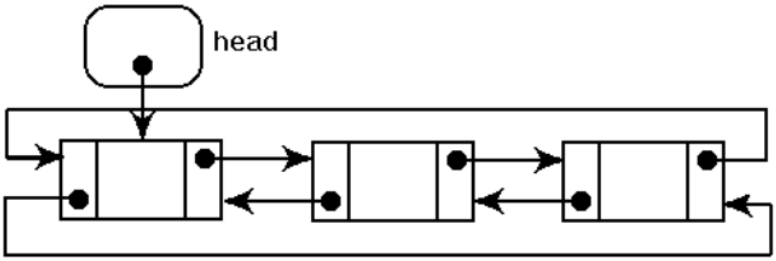
Singly Circular Linked Lists :

The last node points the first node.



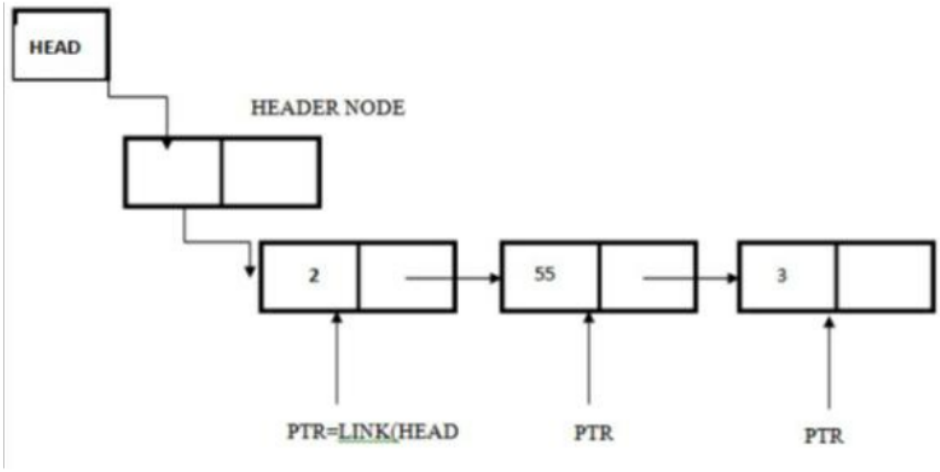
Circular Linked Lists :

The last node points to the first node and the first node points to the last.

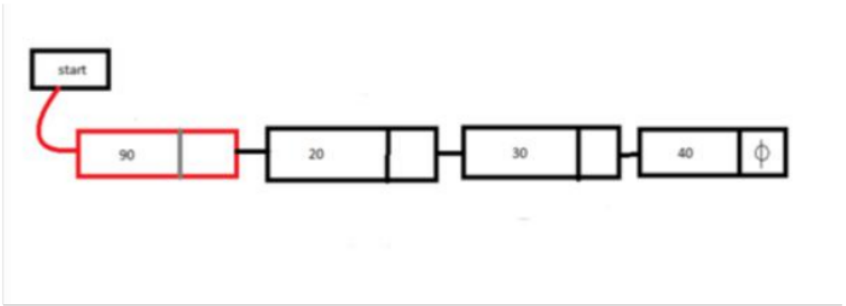


Doubly Linked Circular list

4. Header Linked List (HLL)



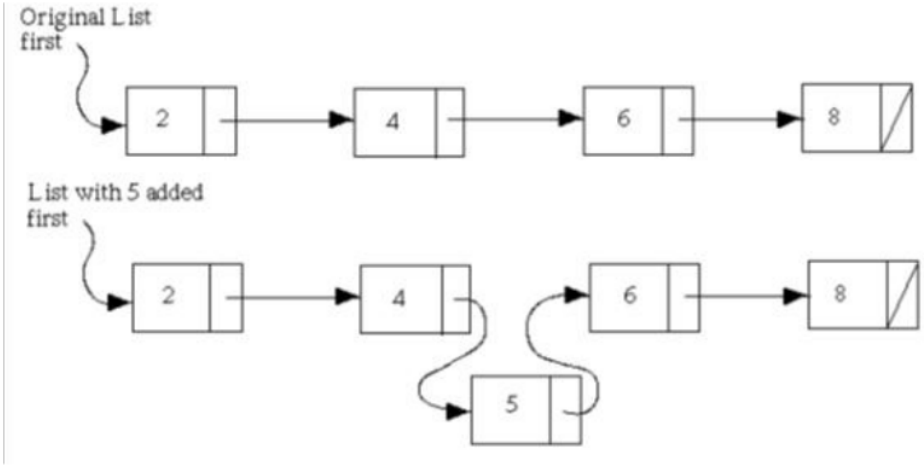
Header linked list is a linked list in which an extra node called the header node is present between the head node and the first node of the linked list.
The header node does not need to store any data. It can be used as a reference node or be left blank.
The header node can also contain the sum of all the elements stored in the linked list.
Header node (in red) contains 90 which is sum of all other elements.



Operations on Linked Lists :

INSERTION

1. **Insertion at beginning:** A node can be added in the beginning of the List. In this, the head points to the new node and the new node is made to point to the node to which the address previously pointed.
2. **Insert at end**: In this, we traverse up to the last node and then the node at the end is made to point to the new node and the new node is pointed to NULL.
3. **Insertion at kth position:** In this we traverse up to the node at k-1 and make this node point to the new node and make the new node point to the node at which the (k-1) th node pointed.
4. **Insertion before/after key:** In this we traverse up to the node having the key value and according to the condition insert the new node before or after that node.



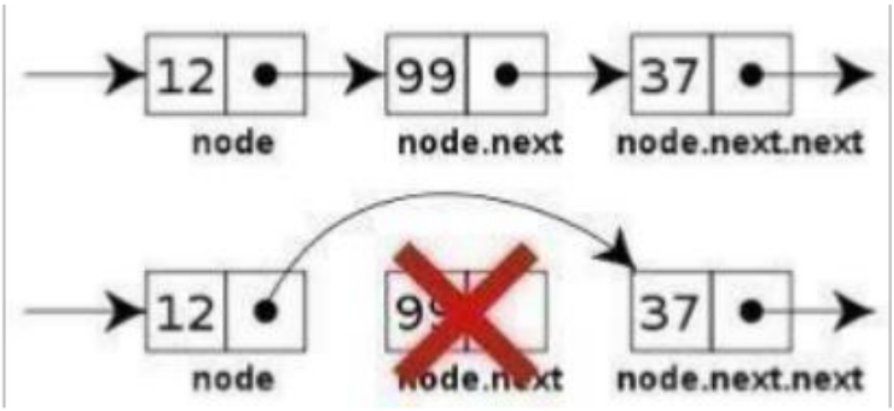
DELETION

1. Deletion at beginning: In this we delete the node to which the head points and head is now made to point the the node at which the deleted node pointed.

2. Deletion at end: In this we traverse up to the second last node and make that node point to NULL and delete the last node i.e. the node to which this node points.

3. Deletion at kth position: In this we traverse up to the node at k-1 position and make it point to the node at k+1 position and delete the node to which the (k-1) th node previously pointed.

4. Deletion before/after key: In this we traverse up to the node having the key value and according to the condition delete the node before or after that node.



SEARCH:

Searching Algorithms are designed to check for an element or retrieve an element from any data structure where it is stored. Based on the type of search operation, these algorithms are generally classified into two categories:

1.

Sequential Search: In this, the list or array is traversed sequentially and every element is checked. For example: Linear Search.

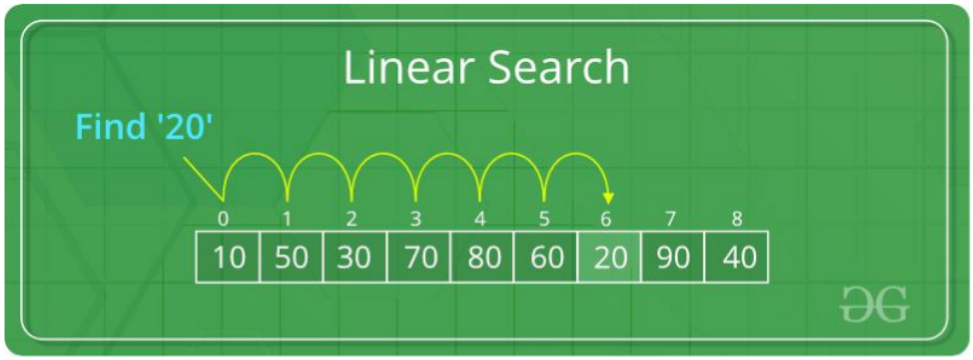
2.

1.

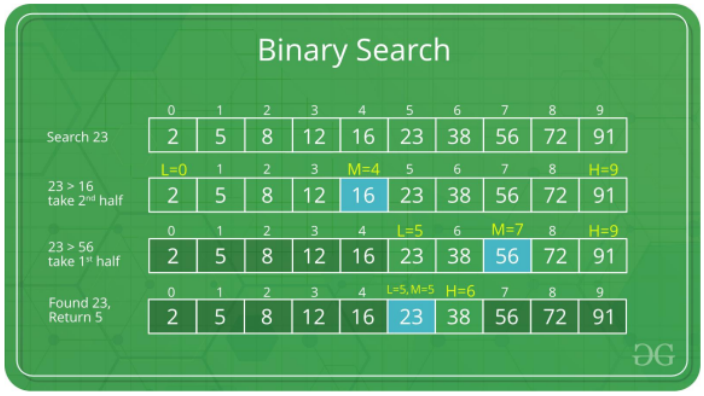
Interval Search: These algorithms are specifically designed for searching in sorted data-structures. These type of searching algorithms are much more efficient than Linear Search as they repeatedly target the center of the search structure and divide the search space in half. For Example: Binary Search.

2.

Linear Search to find the element “20” in a given list of numbers



Binary Search to find the element “23” in a given list of numbers



CHAPTER 4

APPLICATION

- This project can be implemented in any company by fulfilling basic requirements. This Inventory Management System will provide the information about the incoming and outgoing of products inside the factory without manual processing. All information will be updated automatically by using the information stored in the system files.
- Maintain records for long time.
- Fast and perfect.
- Time Management.
- Automatic upgradation of Information.

CHAPTER 5

CONCLUTION

Inventory Management System would be able to maintain information and able to keep records of that particular event. This project can be implemented in any Company by fulfilling basic requirements. This Inventory Management System will provide the information about the incoming and outgoing of products inside the factory without manual processing. All information will be updated automatically by using the information stored in the system files.

5.1: SOURCE CODE

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <string.h>

int add_product();
int delete_product();
void display_product();
void modify_product();
void inventory();

struct node
{
int product_no;
char product_name[10];
int no_of_product;
struct node *next;
};
struct node *head;

int main()
{
int pass;
printf("\n\n\n*_*     "'WELCOME'"     *_*\n");
printf("\n\n\nINVENTORY MANAGEMENT\n");
printf("\n\nHII USER ,YOU REQUIRE A PASSWORD TO GET ACCESS TO THIS PROJECT\n");
```

```
printf("\nGIVE A TRY *_* :");
scanf("%d", &pass);
if (pass == 560103)
{
inventory();
}
else
{
printf("clue :New horizon college of Engineering\n");
printf("clue 2:Pincode of This Location\n");
printf("GIVE A TRY NOW !!!!");
printf("ENTER PASSWORD::");
scanf("%d", &pass);
if (pass == 560103)
{
inventory();
}
else
{
printf("SORRY YOUR CHANCE IS OVER ,YOU CANNOT ACCESS TO THIS PROJECT\n");
}
}
return 0;
}


void inventory()
{
char ch;
```

```
int choice, i, get;

do

{

printf("\n\n\n\t\t\t\t\t");

printf("\nInventory Management System\n");

printf("\n\n\t\tSELECT YOUR CHOICE.....\n\n\t\t\t\t1=ADD PRODUCT\n\t\t\t\t2=DELETE
PRODUCT\n\t\t\t\t3=DISPLAY PRODUCT \n\t\t\t\t4= MODIFY PRODUCT\n\t\t\t\t5=Exit\n\n");

scanf("%d", &choice);

switch (choice)

{

case 1:

get = add_product();

break;

case 2:

get = delete_product();

break;

case 3:

display_product();

break;

case 4:

modify_product();

break;

case 5:

break;

default:

printf("\nINVALID CHOICE ,PLEASE SELECT CORRECT OPTION\n");

}



} while (choice != 5);
```

```
printf("\n\n\n\t\t\t*_* THANK YOU *_*\n");
printf("\n\n\n\t\t\t\t\t\t\"PROJECT DONE BY -- MANIKANTA P,USN--1NH20CS125\n");
printf("\t\t\t\t\t\tHAVE A GOOD DAY\n");
printf("\n\n\n\t\t\t*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*\n");




getch();

}



int add_product()

{

struct node *temp;

struct node *p;

int choice;

int pos = 0;

int count = 0;

char product[30];

p = head;

do

{

struct node *p, *temp;

p = head;

printf("\n\n\tSELECT YOUR CHOICE.....\n\n\t\t\t\t1=ADD AT FIRST\n\t\t\t\t2=ADD AT LAST\n\t\t\t\t3=ADD
IN BETWEEN\n\t\t\t\t5=Exit\n\n");

scanf("%d", &choice);

switch (choice)

{

case 1:
```

```
if (head == NULL)
{   printf("LIST IS EMPTY");
temp = (struct node *)malloc(sizeof(struct node));
printf("\nENTER PRODUCT NUMBER:#");
scanf("%d", &temp->product_no);
printf("\nENTER PRODUCT NAME--");
scanf("%s", &temp->product_name);
printf("\nENTER NO OF PRODUCTS COUNT:#");
scanf("%d", &temp->no_of_product);
head = temp;
temp->next = NULL;
printf("\n\t\t ......INVENTORY HAS OPENED AND THE PRODUCT IS ADDED.....\n");
break;
}
else
{
temp = (struct node *)malloc(sizeof(struct node));
printf("\nENTER PRODUCT NUMBER:#");
scanf("%d", &temp->product_no);
printf("\nENTER PRODUCT NAME--");
scanf("%s", &temp->product_name);
printf("\nENTER NO OF PRODUCTS COUNT:#");
scanf("%d", &temp->no_of_product);
temp->next = head;
head = temp;
printf("\n\t\t ......THE PRODUCT IS ADDED AT FIRST PLACE SUCCESSFULLY.....\n");
break;
}
case 2:
```

```
if (head == NULL)
{
temp = (struct node *)malloc(sizeof(struct node));
printf("\nENTER PRODUCT NUMBER:#");
scanf("%d", &temp->product_no);
printf("\nENTER PRODUCT NAME--");
scanf("%s", &temp->product_name);
printf("\nENTER NO OF PRODUCTS COUNT:#");
scanf("%d", &temp->no_of_product);
head = temp;
temp->next = NULL;
printf("\n\t\t ......INVENTORY HAS OPENED AND THE PRODUCT IS ADDED.....\n");
break;
}
else
{
while (p->next != NULL)
{
p = p->next;
}
temp = (struct node *)malloc(sizeof(struct node));
printf("\nENTER PRODUCT NUMBER:#");
scanf("%d", &temp->product_no);
printf("\nENTER PRODUCT NAME--");
scanf("%s", &temp->product_name);
printf("\nENTER NO OF PRODUCTS COUNT:#");
scanf("%d", &temp->no_of_product);
p->next = temp;
temp->next = NULL;
```

```
printf("\n\t\t ......THE PRODUCT IS ADDED AT LAST SUCCESSFULLY.....\n");
}
break;

case 3:
printf("\n\tENTER THE POSITION,THAT YOU WISH TO ADD AT =");
scanf("%d", &pos);
while (p != NULL)
{
count += 1;
if ((count + 1) == pos)
{
temp = (struct node *)malloc(sizeof(struct node));
printf("\nENTER PRODUCT NUMBER:#");
scanf("%d", &temp->product_no);
printf("\nENTER PRODUCT NAME--");
scanf("%s", &temp->product_name);
printf("\nENTER NO OF PRODUCTS COUNT:#");
scanf("%d", &temp->no_of_product);
temp->next = p->next;
p->next = temp;
printf("\n\t\t ......PRODUCT ADDED SUCCESSFULLY.....\n");
getch();
}
p = p->next;
}
break;
}
```

```
}while (choice != 5);
}

void display_product()
{
struct node *p;
p = head;
if (p == NULL)
{
printf("\n\t\t.... INVENTORY IS EMPTY....\n\t\t....ADD PRODUCT TO SEE THE INVENTORY....\n");
return;
}
printf("\t\t\t**********INVENTORY MANAGEMENT SYSTEM**********");
printf("\n-----------------------------------------------------------------\n");
printf("Product Number | Product Name | Product count");
printf("\n-----------------------------------------------------------------\n");
while (p != NULL)
{
printf("%d",p->product_no);
printf("\t\t%s",p->product_name);
printf("\t       %d",p->no_of_product);
printf("\n");

p = p->next;
}
}

int delete_product() // funtion to delete the data
```

```
{
int choice = 0, count = 0;
char product[30];
do
{
struct node *p, *temp;


if (head == NULL)
{
printf("\n\t\t.... INVENTORY IS EMPTY....\n\t\t....ADD PRODUCT TO DELETE THE PRODUCTS IN
INVENTORY....\n");
return 0;
}
p = head;
printf("\n\n\t\tSELECT YOUR CHOICE.....\n\n\t\t\t\t1=DELETE AT FIRST\n\t\t\t\t\t 2=DELETE AT LAST\n\t\t\t\t\t
3=DELETE IN BETWEEN\n\t\t\t\t\t 5=Exit\n\n");
scanf("%d", &choice);
switch (choice)
{
case 1:temp = head;
head = temp->next;
free(temp);
printf("\n\t\t....FIRST PRODUCT INFO DELETED....\n");
break;
case 2:
if (head->next == NULL)
{
temp = head;
head = temp->next;
```

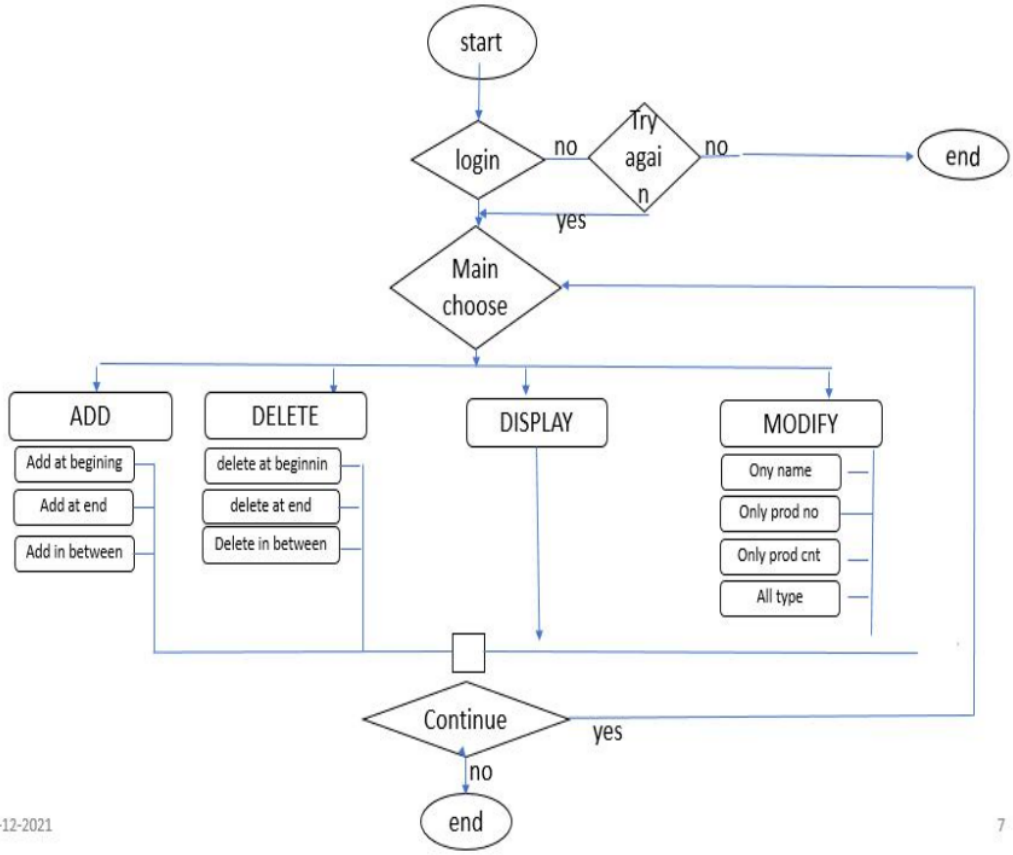
```
free(temp);
printf("\n\t\t....FIRST PRODUCT INFO DELETED....\n");
break;
}
while (p->next->next != NULL)
{
p = p->next;
}
temp = p->next;
p->next = NULL;
free(temp);
printf("\n\t\t....PRODUCT INFO DELETED...\n");
break;
case 3:
printf("\nENTER THE PRODUCT TO BE DELETED=");
scanf("%s", product);
if (strcmp(head->product_name, product) == 0)
{
temp = head;
head = temp->next;
free(temp);
printf("\n\t\t....FIRST PRODUCT INFO DELETED....\n");
break;
}
while (p->next != NULL)
{
if (strcmp(p->next->product_name, product) == 0)
{
count = 1;
```

```
temp = p->next;
p->next = temp->next;
free(temp);
printf("\n\t\t....PRODUCT %s IS REMOVED FROM INVENTORY....\n", product);
break;
}
p = p->next;
}
if (count == 0)
printf("\n\t\t.....NOT FOUND....\n");
break;
}
} while (choice != 5);
return 0;
}
void modify_product()
{ struct node *p, *temp;
int count=0, choice;
char ch[20];
p=head;
if(head==NULL)
{ printf("\n\t\t.... INVENTORY IS EMPTY....\n\t\t....ADD PRODUCT TO MODIFY....\n");
getch();
return;
}
printf("\n\t\t\tENTER THE PRODUCT NAME ,THAT YOU WISH TO EDIT\n");
scanf("%s",&ch);
while(p!=NULL)
{  if(strcmp(p->product_name,ch)==0)
```

```
{ count=1;

printf("\n\t\t\t\t1.EDIT PRODUCT NAME\n\t\t\t\t2.EDIT PRODUCT NUMBER\n\t\t\t\t3.EDIT PRODUCT
COUNT\n\t\t\t\t4.EDIT ALL PRODUCT DETAILS\n\t\t\t\t5.EXIT");
2
scanf("%d",&choice);

switch(choice)

{case 1:printf("\n ENTER NEW PRODUCT NAME:");

scanf("%s",&p->product_name);

break;

case 2:printf("\nENTER NEW PRODUCT NO#:");

scanf("%d",&p->product_no);

break;

case 3:printf("\nENTER NEW NO OF PRODUCTS COUNT#:");

scanf("%d",&p->no_of_product);

break;


case 4: printf("\n ENTER NEW PRODUCT NAME:");

scanf("%s",&p->product_name);

printf("\nENTER NEW PRODUCT NO:#");

scanf("%d",&p->product_no);

printf("\nENTER NEW NO OF PRODUCTS COUNT#:");

scanf("%d",&p->no_of_product);




break;

case 5:break;

default:printf("\n\nINVALID CHOICE !!!!!");

}

printf("\n\t\t\t\t...SUCCESSFULLY MODIFIED...\n");

}
```

```
p=p->next;
}
getch();
if(count==0)
{ printf("\t\t.....NOT FOUND.....\n");
getch();
}
```

FLOW CHART:



SAMPLE OUTPUT:

- Output
- When we run we get this interface,(pass==560103)

```
INVENTORY MANAGEMENT

HII USER ,YOU REQUIRE A PASSWORD TO GET ACCESS TO THIS PROJECT
GIVE A TRY *_* :560103


Inventory Management System

                SELECT YOUR CHOICE.........

                                        1=ADD PRODUCT
                                        2=DELETE PRODUCT
                                        3=DISPLAY PRODUCT
                                        4= MODIFY PRODUCT
                                        5=Exit
```


- Addtion funtion output

```
LIST IS EMPTY
ENTER PRODUCT NUMBER:#01
ENTER PRODUCT NAME--pencil
ENTER NO OF PRODUCTS COUNT:#100
                .....INVENTORY HAS OPENED AND THE PRODUCT IS ADDED......

                SELECT YOUR CHOICE.........

                                        1=ADD AT FIRST
                                        2=ADD AT LAST
                                        3=ADD IN BETWEEN
                                        5=Exit
1
ENTER PRODUCT NUMBER:#02
ENTER PRODUCT NAME--pen
ENTER NO OF PRODUCTS COUNT:#200
                .....THE PRODUCT IS ADDED AT FIRST PLACE SUCCESSFULLY......

                SELECT YOUR CHOICE.........

                                        1=ADD AT FIRST
                                        2=ADD AT LAST
                                        3=ADD IN BETWEEN
                                        5=Exit
```

- Deletion function output

```
Inventory Management System


               SELECT YOUR CHOICE.........

                                    1=ADD PRODUCT
                                    2=DELETE PRODUCT
                                    3=DISPLAY PRODUCT
                                    4= MODIFY PRODUCT
                                    5=Exit

2


               SELECT YOUR CHOICE.........

                                    1=DELETE AT FIRST
                                     2=DELETE AT LAST
                                     3=DELETE IN BETWEEN
                                     5=Exit

1

               ....FIRST PRODUCT INFO DELETED....
```

- Display function output:

```
 Inventory Management System


                SELECT YOUR CHOICE.........

                                     1=ADD PRODUCT
                                     2=DELETE PRODUCT
                                     3=DISPLAY PRODUCT
                                     4= MODIFY PRODUCT
                                     5=Exit

 3
                      **********INVENTORY MANAGEMENT SYSTEM**********
 ------------------------------------------------------------
 Product Number | Product Name | Product count
 ------------------------------------------------------------
 1              pencil         100
```

- Modify function output

```
3
                    ******************INVENTORY MANAGEMENT SYSTEM*********************
-------------------------------------------------------------
Product Number | Product Name | Product count
-------------------------------------------------------------
1              pencil         100


Inventory Management System


               SELECT YOUR CHOICE.........

                                        1=ADD PRODUCT
                                        2=DELETE PRODUCT
                                        3=DISPLAY PRODUCT
                                        4= MODIFY PRODUCT
                                        5=Exit

4

                        ENTER THE PRODUCT NAME ,THAT YOU WISH TO EDIT
pencil

                                    1.EDIT PRODUCT NAME
                                    2.EDIT PRODUCT NUMBER
                                    3.EDIT PRODUCT COUNT
                                    4.EDIT ALL PRODUCT DETAILS
                                    5.EXIT1

 ENTER NEW PRODUCT NAME:newproduct

                                      SUCCESSFULLY MODIFIED
```

5.3 REFERENCES:

- www.geeksforgeeks.org/data-structures/linked-list/
- www.udemy.com/course/datastructurescncpp/learn/lecture/

1NH20CS125

ORIGINALITY REPORT

10%	%	10%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	S.N. Sivanandam, S.N. Deepa. "Chapter 9 Genetic Algorithm Optimization in C/C++", Springer Science and Business Media LLC, 2008 Publication	2%
2	"Question Papers and Solutions: DC-05 (Problem Solving through 'C')", IETE Journal of Education, 2015 Publication	2%
3	Elshad Karimov. "Data Structures and Algorithms in Swift", Springer Science and Business Media LLC, 2020 Publication	1%
4	Kupferschmid, . "Memory Management Techniques", Classical Fortran, 2009. Publication	1%
5	Patrícia Almeida, Luísa Agante. "Comparing consumer decision skills in institutionalized vs family children", Journal of Consumer Marketing, 2016 Publication	1%

6	S. García-Ferreira, C. Yescas-Aparicio. "Families of retractions and families of closed subsets on compact spaces", Topology and its Applications, 2021 Publication	1 %
7	Everardo Reyes-Garcia. "Describing Graphical Information", Wiley, 2017 Publication	1 %
8	R.P. Broadwater, J.C. Thompson, R.E. Lee, H. Maghdan-D. "Computer-aided protection system design with reconfiguration", IEEE Transactions on Power Delivery, 1991 Publication	<1 %
9	Wenchao Zhou, Qiong Fei, Arjun Narayan, Andreas Haeberlen, Boon Thau Loo, Micah Sherr. "Secure network provenance", Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles - SOSP '11, 2011 Publication	<1 %
10	"Informatik für Ingenieure und Naturwissenschaftler", Springer Science and Business Media LLC, 2006 Publication	<1 %
11	Xingni Zhou, Qiguang Miao, Lei Feng. "Programming in C", Walter de Gruyter GmbH, 2020 Publication	<1 %

12	Bill Buchanan. "Handbook of Data Communications and Networks", Springer Science and Business Media LLC, 1999 Publication	<1%
13	E. G. Daylight, S. Wuytack, C. Ykman-Couvreur, F. Catthoor. "Analyzing energy friendly steady state phases of dynamic application execution in terms of sparse data structures", Proceedings of the 2002 international symposium on Low power electronics and design - ISLPED '02, 2002 Publication	<1%
14	Namrata Vaswani. "Stability (over time) of modified-CS for recursive causal sparse reconstruction", 2010 48th Annual Allerton Conference on Communication, Control, and Computing (Allerton), 2010 Publication	<1%

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	On		