**Digital Assignment- 2**

**Data Structures (Lab)**

Submitted by: Hari Krishna Shah

VIT ID: 21BCS0167

Link:<https://drive.google.com/drive/folders/19gQ6qLNeWQu3_6CzPHfKICQrybHrqRk0?usp=sharing>

Note: All programs can be downloaded from the link above.

Ques 1. Implement single linked list operation such as 1. Insertion at front, 2. Insertion in middle 3 insertion at the last 4. Deletion at front, 5. Deletion at last 6. deletion in the middle. 7. Sorting 8. Counting nodes.

Answer:

#include <stdio.h>

#include <malloc.h>

*//Coded By Hari Krishna Shah*

**struct** node**{**

**int** data**;**

**struct** node **\***next**;**

**};**

**struct** node **\***create\_and\_expand**(struct** node **\*);**

**struct** node **\***insert\_end**(struct** node **\*);**

**struct** node **\***display**(struct** node **\*);**

**struct** node **\***count**(struct** node **\*);**

**struct** node **\***search**(struct** node **\*);**

**struct** node **\***add\_head**(struct** node **\*);**

**struct** node **\***add\_after**(struct** node **\*);**

**struct** node **\***add\_before**(struct** node **\*);**

**struct** node **\***delete\_head**(struct** node **\*);**

**struct** node **\***delete\_end**(struct** node **\*);**

**struct** node **\***delete\_with\_data**(struct** node **\*);**

**struct** node **\***delete\_before\_data**(struct** node **\*);**

**struct** node **\***delete\_after\_data**(struct** node **\*);**

**struct** node **\***delete\_all**(struct** node **\*);**

**struct** node **\***sort\_acc**(struct** node **\*);**

**struct** node **\***sort\_des**(struct** node **\*);**

**struct** node **\***add\_in\_search**(struct** node **\*);**

**struct** node **\***add\_in\_middle**(struct** node **\*);**

**struct** node **\***delete\_the\_middle**(struct** node **\*);**

**struct** node **\***delete\_except**(struct** node **\*);**

**int** main**(){**

**int** option **=** 0**;**

**struct** node **\***start **=** NULL**;**

**while(**option **!=** **-**1**){**

printf**("\t\t\tThis program is created by Hari Krishna Shah.\n");**

printf**("Welcome to the Main-Menu !\n\**

**Enter 1 to create and expand linked list \n\**

**Enter 2 to add node after end.\n\**

**Enter 3 to display linked list.\n\**

**Enter 4 to count the number of nodes in the linked list.\n\**

**Enter 5 to search the linked list\n \**

**Enter 6 to add node in the beginning\n \**

**Enter 7 to add node after data\n \**

**Enter 8 to add node before data\n \**

**Enter 9 to delete the first node\n \**

**Enter 10 to delete the last node\n \**

**Enter 11 to delete the node with the searched data\n \**

**Enter 12 to delete the node which is before the searched data.\n \**

**Enter 13 to delete the node which is after the searched data\n \**

**Enter 14 to delete the entire list\n \**

**Enter 15 to sort the nodes in accesending ordr\n \**

**Enter 16 to sort the nodes in decending order\n \**

**Enter 17 to add a new data to a data node\n \**

**Enter 18 to add a node in the middle\n \**

**Enter 19 to delete the middle node \n \**

**Enter 20 to delete all the list expect a specified node\n \**

**Enter -1 to exit the program\n");**

printf**("\nEnter your option here: ");**

scanf**("%d",** **&**option**);**

**switch(**option**){**

**case** 1**:**

start **=** create\_and\_expand**(**start**);**

**break;**

**case** 2**:**

start **=** insert\_end**(**start**);**

**break;**

**case** 3**:**

start **=** display**(**start**);**

**break;**

**case** 4**:**

start **=** count**(**start**);**

**break;**

**case** 5**:**

start **=** search**(**start**);**

**break;**

**case** 6**:**

start **=** add\_head**(**start**);**

**break;**

**case** 7**:**

start **=** add\_after**(**start**);**

**break;**

**case** 8**:**

start **=** add\_before**(**start**);**

**break;**

**case** 9**:**

start **=** delete\_head**(**start**);**

**break;**

**case** 10**:**

start **=** delete\_end**(**start**);**

**break;**

**case** 11**:**

start **=** delete\_with\_data**(**start**);**

**break;**

**case** 12**:**

start **=** delete\_before\_data**(**start**);**

**break;**

**case** 13**:**

start **=** delete\_after\_data**(**start**);**

**break;**

**case** 14**:**

start **=** delete\_all**(**start**);**

**break;**

**case** 15**:**

start **=** sort\_acc**(**start**);**

**break;**

**case** 16**:**

start **=** sort\_des**(**start**);**

**break;**

**case** 17**:**

start **=** add\_in\_search**(**start**);**

**break;**

**case** 18**:**

start **=** add\_in\_middle**(**start**);**

**break;**

**case** 19**:**

start **=** delete\_the\_middle**(**start**);**

**break;**

**case** 20**:**

start **=** delete\_except**(**start**);**

**break;**

**case** **-**1**:**

**break;**

**default:**

printf**("Please enter a valid option and try again !!!\n\n");**

**}**

**}**

printf**("Thank you for using the program.\n");**

printf**("The program exited successfully.\n");**

**return** 0**;**

**}**

**struct** node **\***create\_and\_expand**(struct** node **\***start**){**

**struct** node **\***new\_node**,** **\***ptr**,** temp**;**

**int** num**;**

**if(**start **==** NULL**){**

new\_node **=** **(struct** node **\*)** **(**malloc**(sizeof(struct** node**)));**

start **=** new\_node**;**

printf**("Enter the data to insert in the first node: ");**

scanf**("%d",** **&**num**);**

start**->**data **=** num**;**

start**->**next **=** NULL**;**

printf**("The linked list has been created.\n\n");**

**}**

**else{**

printf**("The linked list already exists.\n");**

**}**

printf**("Enter the data below to expand it.\n");**

ptr **=** start**;**

**while(**ptr**->**next **!=**NULL**){**

ptr **=** ptr**->**next**;**

**}**

printf**("Enter -1 to stop expanding the list else enter the data to store in the next node: ");**

scanf**("%d",** **&**num**);**

**while(**num **!=-**1**){**

**struct** node **\***new\_node**;**

new\_node **=** **(struct** node **\*)** **(**malloc**(sizeof(**node**)));**

ptr**->**next **=** new\_node**;**

new\_node**->**next **=** NULL**;**

new\_node**->**data **=** num**;**

ptr **=** new\_node**;**

printf**("Enter -1 to stop expanding the list else enter the data to store in the next node: ");**

scanf**("%d",** **&**num**);**

**}**

printf**("\n");**

**return** start**;**

**}**

**struct** node **\***insert\_end**(struct** node **\***start**){**

**int** num**;**

**struct** node **\***ptr**,** **\***new\_node**;**

new\_node **=** **(struct** node **\*)** **(**malloc**(sizeof(struct** node**)));**

ptr **=** start**;**

printf**("Enter the data to store at the last node: ");**

scanf**("%d",** **&**num**);**

new\_node**->**data **=** num**;**

**if(**start **==** NULL**){**

start **=** new\_node**;**

start**->**next **=** NULL**;**

**return** start**;**

**}**

**while(**ptr**->**next **!=** NULL**){**

ptr **=** ptr**->**next**;**

**}**

ptr**->**next **=** new\_node**;**

new\_node**->**next**=**NULL**;**

printf**("The node has been added at the end.\n\n");**

**return** start**;**

**}**

**struct** node **\***display**(struct** node **\***start**){**

**if(**start **==** NULL**){**

printf**("The linked list is empty.\n\n");**

**return** start**;**

**}**

**struct** node **\***ptr**;**

ptr **=** start**;**

printf**("The data are: ");**

**while(**ptr **!=** NULL**){**

printf**("%d ",** ptr**->**data**);**

ptr **=** ptr**->**next**;**

**}**

printf**("\n\n");**

**return** start**;**

**}**

**struct** node **\***count**(struct** node **\***start**){**

**struct** node **\***ptr**;**

ptr **=** start**;**

**int** count **=** 0**;**

**while(**ptr **!=** NULL**){**

count **+=** 1**;**

ptr **=** ptr**->**next**;**

**}**

printf**("The number of nodes are %d.\n\n",** count**);**

**return** start**;**

**}**

**struct** node **\***search**(struct** node **\***start**){**

**if(**start **==** NULL**){**

printf**("The linked list is empty. Add some nodes first.\n");**

**return** start**;**

**}**

**struct** node **\***ptr**;**

ptr **=** start**;**

**int** i **=** 0**;**

**int** count **=** 0**;**

**int** number**;**

printf**("Enter a number to check in the linked list: ");**

scanf**("%d",** **&**number**);**

**while(**ptr**!=** NULL**){**

**if(**ptr**->**data **==** number**){**

printf**("The number %d is available in the linked list at index %d.\n\n",** number**,** i**);**

count **=** count **+** 1**;**

**}**

i **=** i**+**1**;**

ptr **=** ptr**->**next**;**

**}**

**if(**count **==** 0**){**

printf**("The number %d is not available in the linked list.\n\n",** number**);**

**}**

**return** start**;**

**}**

**struct** node **\***add\_head**(struct** node **\***start**){**

**struct** node **\***new\_node**,** **\***temp**;**

**int** num**;**

temp **=** start**;**

new\_node **=** **(struct** node **\*)** **(**malloc**(sizeof(struct** node**)));**

printf**("Enter the data to insert in the first node: ");**

scanf**("%d",** **&**num**);**

**if(**start **==** NULL**){**

start **=** new\_node**;**

start**->**data **=** num**;**

start**->**next **=** NULL**;**

**}**

**else{**

new\_node**->**next **=** temp**;**

new\_node**->**data **=** num**;**

start **=** new\_node**;**

**}**

printf**("The data has been added to the head of the linked list.\n\n");**

**return** start**;**

**}**

**struct** node **\***add\_after**(struct** node **\***start**){**

**if(**start **==** NULL**){**

printf**("The link is empty. Add some nodes to the list first.\n\n");**

**return** start**;**

**}**

**struct** node **\***new\_node**,** **\***ptr**,** **\***temp**;**

new\_node **=** **(struct** node **\*)** **(**malloc**(sizeof(struct** node**)));**

**int** num**,** check\_data**;**

printf**("Enter the data after which the node is to be added: ");**

scanf**("%d",** **&**check\_data**);**

ptr **=** start**;**

**while(**ptr**->**data **!=** check\_data**){**

ptr **=** ptr**->**next**;**

**if(**ptr **==** NULL**){**

printf**("The searched data couldn't be found. The process terminated.\n\n");**

**return** start**;**

**}**

**}**

printf**("Enter the data to be stored: ");**

scanf**("%d",** **&**num**);**

temp **=** ptr**->**next**;**

ptr**->**next **=** new\_node**;**

new\_node**->**data **=** num**;**

new\_node**->**next **=** temp**;**

printf**("The node has been added after the data %d.\n\n",** check\_data**);**

**return** start**;**

**}**

**struct** node **\***add\_before**(struct** node **\***start**){**

**if(**start **==** NULL**){**

printf**("The link is empty. Add some nodes to the list first.\n\n");**

**return** start**;**

**}**

**struct** node **\***new\_node**,** **\***ptr**,** **\***temp**;**

**int** num**,** check\_data**;**

printf**("Enter the data before which node is to be added: ");**

scanf**("%d",** **&**check\_data**);**

ptr **=** start**;**

**if(**start**->**data **==** check\_data**){**

printf**("The matched data was found to be in the first node of data.\n");**

printf**("Now we need to add a new head to our linked list.\n");**

start **=** add\_head**(**start**);**

**return** start**;**

**}**

**else{**

**while(**ptr**->**data **!=** check\_data**){**

ptr **=** ptr**->**next**;**

**if(**ptr **==** NULL**){**

printf**("The searched data couldn't be found. The process terminated.\n\n");**

**return** start**;**

**}**

**}**

new\_node **=** **(struct** node **\*)** **(**malloc**(sizeof(struct** node**)));**

printf**("Enter the data to be stored: ");**

scanf**("%d",** **&**num**);**

temp **=** ptr**;**

ptr **=** start**;**

**while(**ptr**->**next **!=** temp**){**

ptr **=** ptr**->**next**;**

**}**

ptr**->**next **=** new\_node**;**

new\_node**->**data **=** num**;**

new\_node**->**next **=** temp**;**

**}**

printf**("The node has been added before the data %d.\n\n",** check\_data**);**

**return** start**;**

**}**

**struct** node **\***delete\_head**(struct** node **\***start**){**

**struct** node **\***temp1**,** **\***temp2**;**

temp2 **=** start**;**

**if(**start **==** NULL**){**

printf**("The linked list is empty.\n\n");**

**return** start**;**

**}**

temp1 **=** start**->**next**;**

free**(**temp2**);**

start **=** temp1**;**

printf**("The first node has been deleted sucessfully.\n\n");**

**return** start**;**

**}**

**struct** node **\***delete\_end**(struct** node **\***start**){**

**if(**start **==** NULL**){**

printf**("The linked list is empty.\n\n");**

**return** start**;**

**}**

**if(**start**->**next **==** NULL**){**

printf**("Currenlty, there is only one node in the linked list so, the first node is also the last node.\n");**

start **=** delete\_head**(**start**);**

**return** start**;**

**}**

**struct** node **\***ptr**,** **\***temp**;**

ptr **=** start**;**

**while(**ptr**->**next **!=** NULL**){**

ptr **=** ptr**->**next**;**

**}**

temp **=** ptr**;**

ptr **=** start**;**

**while(**ptr**->**next **!=** temp**){**

ptr **=** ptr**->**next**;**

**}**

ptr**->**next **=** NULL**;**

free**(**temp**);**

printf**("The last node has been deleted sucessfully.\n\n");**

**return** start**;**

**}**

**struct** node **\***delete\_with\_data**(struct** node **\***start**){**

**struct** node **\***ptr**,** **\***temp1**,** **\***temp2**;**

ptr **=** start**;**

**if(**start **==** NULL**){**

printf**("The linked list is empty. Add some nodes first and try again.\n");**

**return** start**;**

**}**

**int** num**;**

printf**("Enter the data of the node which you want to delete: ");**

scanf**("%d",** **&**num**);**

**if(**start**->**data **==** num**){**

start **=** delete\_head**(**start**);**

**return** start**;**

**}**

**while(**ptr**->**data **!=** num**){**

ptr **=** ptr**->**next**;**

**if(**ptr **==** NULL**){**

printf**("The searched data was not found in the linked list.");**

**return** start**;**

**}**

**}**

temp1 **=** ptr**;**

temp2 **=** ptr**->**next**;**

ptr **=** start**;**

**while(**ptr**->**next **!=** temp1**){**

ptr **=** ptr**->**next**;**

**}**

ptr**->**next **=** temp2**;**

free**(**temp1**);**

printf**("The node containing the number %d has been deleted successfully.\n\n",** num**);**

**return** start**;**

**}**

**struct** node **\***delete\_before\_data**(struct** node **\***start**){**

**if(**start **==** NULL**){**

printf**("The linked list is empty.\n\n");**

**return** start**;**

**}**

**struct** node **\***ptr**,** **\***temp1**,** **\***temp2**;**

**int** num**;**

printf**("Enter the data of whose before node is to be deleted: ");**

scanf**("%d",** **&**num**);**

**if(**start**->**data **==** num**){**

printf**("The data is in the first node of the linked list.\nso, its before node can't be deleted.\n\n");**

**return** start**;**

**}**

ptr **=** start**;**

**while(**ptr**->**data **!=** num**){**

ptr **=** ptr**->**next**;**

**if(**ptr **==** NULL**){**

printf**("The searched data is not in the list.\n\n");**

**return** start**;**

**}**

**}**

temp1 **=** ptr**;**

ptr **=** start**;**

**while(**ptr**->**next **!=** temp1**){**

ptr **=** ptr**->**next**;**

**}**

temp2 **=** ptr**;**

**if(**temp2 **==** start**){**

start **=** delete\_head**(**start**);**

**return** start**;**

**}**

ptr **=** start**;**

**while(**ptr**->**next **!=** temp2**){**

ptr **=** ptr**->**next**;**

**}**

ptr**->**next **=** temp1**;**

free**(**temp2**);**

printf**("The node before the number %d has been deleted.\n\n",** num**);**

**return** start**;**

**}**

**struct** node **\***delete\_after\_data**(struct** node **\***start**){**

**if(**start **==** NULL**){**

printf**("The linked list is empty.\n\n");**

**return** start**;**

**}**

**struct** node **\***ptr**,** **\***temp1**,** **\***temp2**;**

**int** num**;**

printf**("Enter the data of which suceeding node is to be deleted: ");**

scanf**("%d",** **&**num**);**

ptr **=** start**;**

**while(**ptr**->**data **!=** num**){**

ptr **=** ptr**->**next**;**

**}**

**if(**ptr**->**next **==** NULL**){**

printf**("The matched data is in the last node so its succedding node cant be deleted.\n\n");**

**return** start**;**

**}**

temp1 **=** ptr**;**

temp2 **=** temp1**->**next**;**

ptr **=** temp2**->**next**;**

temp1**->**next **=** ptr**;**

free**(**temp2**);**

printf**("The node after the data %d has been deleted.\n\n",** num**);**

**return** start**;**

**}**

**struct** node **\***delete\_all**(struct** node **\***start**){**

**struct** node **\***ptr**,** **\***temp**;**

ptr **=** start**;**

**if(**start **==** NULL**){**

printf**("The linked list is empty.Add some nodes and try again.\n");**

**return** start**;**

**}**

**while(**ptr **!=** NULL**){**

temp **=** ptr**->**next**;**

free**(**ptr**);**

ptr **=** temp**;**

**}**

printf**("The entire linked list has been deleted.\n");**

start **=** NULL**;**

**return** start**;**

**}**

**struct** node **\***sort\_acc**(struct** node **\***start**){**

**struct** node **\***ptr**,** **\***succedptr**;**

**int** temp**;**

ptr **=** start**;**

**int** count **=** 0**;**

**if(**start **==** NULL**){**

printf**("The linked list is empty. Add some nodes first.\n");**

**return** start**;**

**}**

succedptr **=** ptr**->**next**;**

**if(**start**->**next **==** NULL**){**

printf**("The linked has only one element.So, it doesn't need to be sorted.\n");**

**return** start**;**

**}**

**while(**ptr **!=** NULL**){**

succedptr **=** ptr**->**next**;**

**while(**succedptr **!=** NULL**){**

**if(**ptr**->**data**>**succedptr**->**data**){**

temp **=** ptr**->**data**;**

ptr**->**data **=** succedptr**->**data**;**

succedptr**->**data **=** temp**;**

**}**

succedptr **=** succedptr**->**next**;**

**}**

ptr **=** ptr**->**next**;**

**}**

printf**("The linked list has been sorted ascending order successfully.\n");**

**return** start**;**

**}**

**struct** node **\***sort\_des**(struct** node **\***start**){**

**struct** node **\***ptr**,** **\***succedptr**;**

**int** temp**;**

ptr **=** start**;**

**int** count **=** 0**;**

**if(**start **==** NULL**){**

printf**("The linked list is empty. Add some nodes first.\n");**

**return** start**;**

**}**

succedptr **=** ptr**->**next**;**

**if(**start**->**next **==** NULL**){**

printf**("The linked has only one element.So, it doesn't need to be sorted.\n");**

**return** start**;**

**}**

**while(**ptr **!=** NULL**){**

succedptr **=** ptr**->**next**;**

**while(**succedptr **!=** NULL**){**

**if(**ptr**->**data**<**succedptr**->**data**){**

temp **=** ptr**->**data**;**

ptr**->**data **=** succedptr**->**data**;**

succedptr**->**data **=** temp**;**

**}**

succedptr **=** succedptr**->**next**;**

**}**

ptr **=** ptr**->**next**;**

**}**

printf**("The linked list has been sorted in descending order successfully.\n");**

**return** start**;**

**}**

**struct** node **\***add\_in\_search**(struct** node **\***start**){**

**int** data**;**

**if(**start **==** NULL**){**

printf**("The link is empty. Add some nodes to the list first.\n\n");**

**return** start**;**

**}**

**struct** node **\***ptr**,** **\***succedptr**,** **\***preptr**;**

ptr **=** start**;**

**int** search**;**

printf**("Enter the data to number to search: ");**

scanf**("%d",** **&**search**);**

**while(**ptr**->**data **!=** search**){**

ptr **=** ptr**->**next**;**

**if(**ptr **==** NULL**){**

printf**("The searched data couldn't be found. The process terminated.\n\n");**

**return** start**;**

**}**

**}**

succedptr **=** ptr**->**next**;**

printf**("Enter the new data to be inserted in this node: ");**

scanf**("%d",** **&**data**);**

ptr**->**data **=** data**;**

ptr**->**next **=** succedptr**;**

printf**("The new data has been added to the searched data successfully.\n");**

**return** start**;**

**}**

**struct** node **\***add\_in\_middle**(struct** node **\***start**){**

**if(**start **==** NULL**){**

printf**("The linked list is empty. Add some nodes first.\n\n");**

**return** start**;**

**}**

**struct** node **\***ptr**,** **\***new\_node**;**

new\_node **=** **(struct** node **\*)** **(**malloc**(sizeof(struct** node**)));**

**int** count **=** 0**,** middle**;**

ptr **=** start**;**

**int** data**;**

printf**("Enter the data to store in the middle of the linked list: ");**

scanf**("%d",** **&**data**);**

**while(**ptr **!=** NULL**){**

ptr **=** ptr**->**next**;**

count **+=** 1**;**

**}**

**if(**count **%** 2 **==** 0**){**

middle **=** **(**count**/**2**)-**1**;**

**}**

**else{**

middle **=** **((**count**-**1**)/**2**);**

**}**

ptr **=** start**;**

**for(int** i **=** 0**;** i**<**middle**;** i**++){**

ptr **=** ptr**->**next**;**

**}**

new\_node**->**data **=** data**;**

**if(**start**->**next **==** NULL**){**

start**->**next **=** new\_node**;**

new\_node**->**next **=** NULL**;**

**return** start**;**

**}**

new\_node**->**next **=** ptr**->**next**;**

ptr**->**next **=** new\_node**;**

**return** start**;**

**}**

**struct** node **\***delete\_the\_middle**(struct** node **\***start**){**

**if(**start **==** NULL**){**

printf**("The linked list is empty. Add some nodes first.\n\n");**

**return** start**;**

**}**

**if(**start**->**next **==** NULL**){**

printf**("The linked list has only 1 node so it's middle node can't be deleted.\n");**

**return** start**;**

**}**

**struct** node **\***ptr**,** **\***succedptr**,** **\***mid\_node**;**

**int** count **=** 0**,** middle**;**

ptr **=** start**;**

**while(**ptr **!=** NULL**){**

ptr **=** ptr**->**next**;**

count **+=** 1**;**

**}**

**if(**count **%** 2 **==** 0**){**

middle **=** **(**count**/**2**);**

**}**

**else{**

middle **=** **((**count**-**1**)/**2**);**

**}**

ptr **=** start**;**

**for(int** i **=** 0**;** i**<**middle**;** i**++){**

ptr **=** ptr**->**next**;**

**}**

mid\_node **=** ptr**;**

**if(**mid\_node**->**next **==** NULL**){**

ptr **=** start**;**

**while(**ptr**->**next **!=** mid\_node**){**

ptr **=** ptr**->**next**;**

**}**

ptr**->**next **=** NULL**;**

free**(**mid\_node**);**

**return** start**;**

**}**

succedptr **=** ptr**->**next**;**

ptr **=** start**;**

**while(**ptr**->**next **!=** mid\_node**){**

ptr **=** ptr**->**next**;**

**}**

ptr**->**next **=** succedptr**;**

free**(**mid\_node**);**

**return** start**;**

**}**

**struct** node **\***delete\_except**(struct** node **\***start**){**

**if(**start **==** NULL**){**

printf**("The linked list is empty. Add some nodes first.\n");**

**return** start**;**

**}**

**struct** node **\***ptr**,** temp**,** **\***deltemp**;**

**int** search**;**

ptr **=** start**;**

printf**("Enter the data of the node which is to kept and other nodes are to be deleted: ");**

scanf**("%d",** **&**search**);**

**while(**ptr**->**data **!=** search**){**

ptr **=** ptr**->**next**;**

**if(**ptr **==** NULL**){**

printf**("The searched data was not found in the linked list. Please try with another again data.\n");**

**return** start**;**

**}**

**}**

temp **=** **\***ptr**;**

**while(**ptr **!=** NULL**){**

deltemp **=** ptr**->**next**;**

free**(**ptr**);**

ptr **=** deltemp**;**

**}**

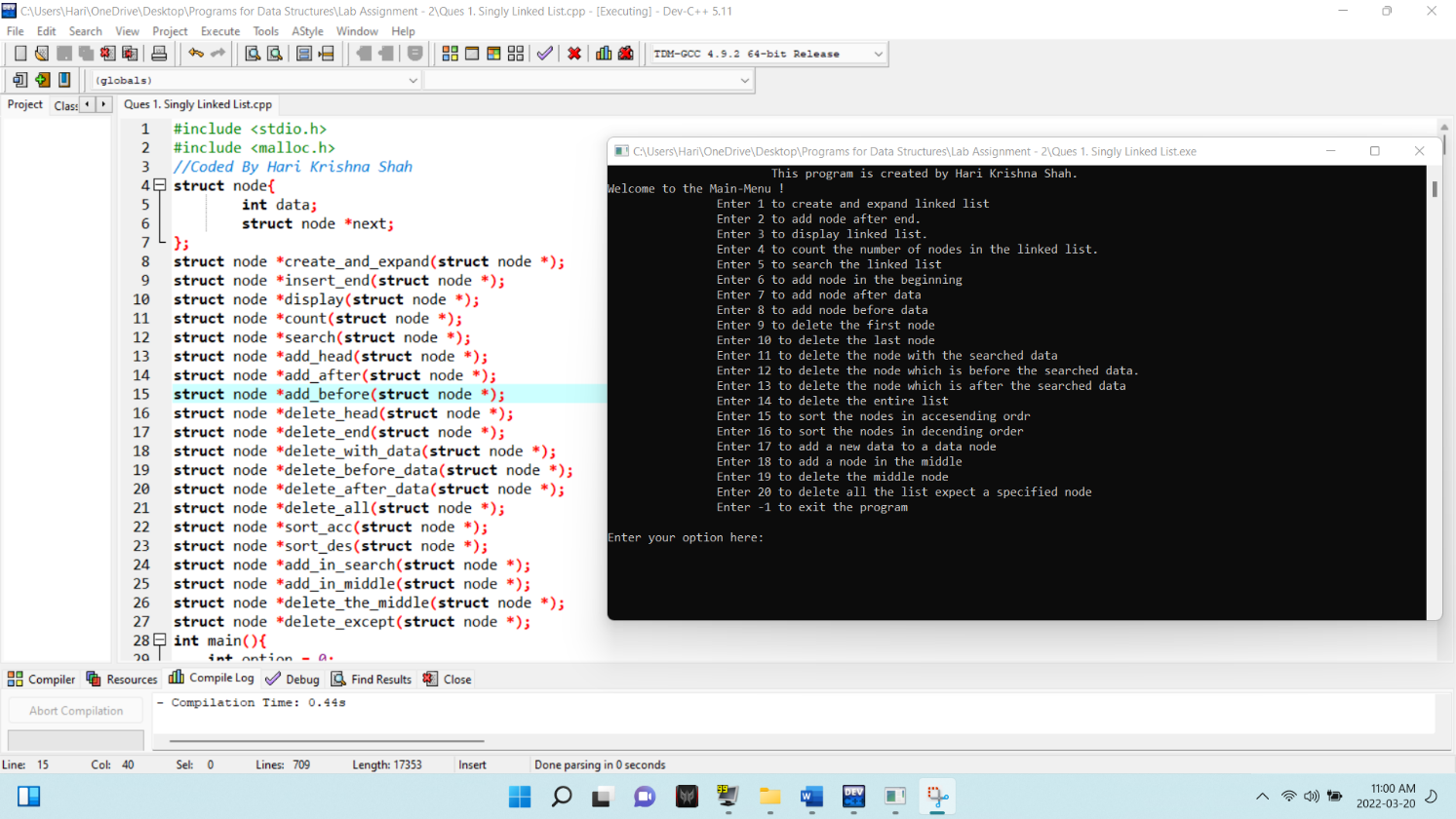
start **=** **(struct** node **\*)** **(** malloc**(sizeof(struct** node **)));**

**\***start **=** temp**;**

start**->**next **=** NULL**;**

**return** start**;**

**}**



Ques 2. Write a program to maintain the records of students in an effective dynamic structure. Search a particular record based on the roll number and display the previous and next values of that node.

Answer:

#include <stdio.h>

#include <malloc.h>

#include <string.h>

*//Coded By Hari Krishna Shah*

**struct** node**{**

**char** name**[**50**];**

**int** roll**;**

**char** address**[**50**];**

**char** course**[**50**];**

**struct** birth\_date**{**

**int** day**;**

**int** month**;**

**int** year**;**

**}**dob**;**

**float** marks**[**5**];**

**float** average**;**

**char** grade**[**5**];**

**float** total**;**

**struct** node **\***previous**;**

**struct** node **\***next**;**

**};**

**struct** node **\***create\_and\_add**(struct** node **\*);**

**struct** node **\***display\_all**(struct** node **\*);**

**struct** node **\***search**(struct** node **\*);**

**struct** node **\***delete\_head**(struct** node **\*);**

**struct** node **\***delete\_end**(struct** node **\*);**

**struct** node **\***delete\_search**(struct** node **\*);**

**struct** node **\***delete\_before**(struct** node **\*);**

**struct** node **\***delete\_after**(struct** node **\*);**

**struct** node **\***delete\_all**(struct** node **\*);**

**struct** node **\***delete\_except**(struct** node **\*);**

**struct** node **\***count**(struct** node **\*);**

**struct** node **\***add\_head**(struct** node **\*);**

**struct** node **\***add\_before**(struct** node **\*);**

**struct** node **\***add\_after**(struct** node **\*);**

**struct** node **\***add\_in\_search**(struct** node **\*);**

**struct** node **\***sort\_ascend**(struct** node **\*);**

**struct** node **\***sort\_descen**(struct** node **\*);**

**struct** node **\***insert\_end**(struct** node **\*);**

**struct** node **\***add\_in\_middle**(struct** node **\*);**

**struct** node **\***delete\_the\_middle**(struct** node **\*);**

**void** display**(struct** node **\*);**

**int** main**(){**

**int** option **=** 0**;**

**struct** node **\***start **=** NULL**;**

**while(**option **!=** **-**1**){**

printf**("\t\t\tThis program is made by Hari Krishna Shah !!!\n");**

printf**("Welcome to the Main Menu.\n");**

printf**("Please choose an option from the menu below\n \**

**Enter 1 to create and add student records to the end of the linked list\n \**

**Enter 2 to display all the student's records\n \**

**Enter 3 to search a student's record\n \**

**Enter 4 to delete the first node\n \**

**Enter 5 to delete the last node\n \**

**Enter 6 to delete the student record of the searched roll number of the student\n \**

**Enter 7 to delete the node before a roll number\n \**

**Enter 8 to delete the node after a roll number\n \**

**Enter 9 to delete the entire student record except a specified a student of specified roll number\n \**

**Enter 10 to delete the entire student record\n \**

**Enter 11 to count the number of nodes\n \**

**Enter 12 to add student record in the first node\n \**

**Enter 13 to add student record before searched roll number\n \**

**Enter 14 to add student record after searched roll number\n \**

**Enter 15 to add or edit student record of the searched roll number \n \**

**Enter 16 to sort the nodes in ascending order according to roll number\n \**

**Enter 17 to sort the nodes in descending order according to roll number\n \**

**Enter 18 to add student record at the end of the linked list\n \**

**Enter 19 to add a new student record at the middle node\n \**

**Enter 20 to delete the student record at the middle node\n \**

**Enter -1 to quit the program\n");**

printf**("Enter your option here: ");**

scanf**("%d",** **&**option**);**

printf**("\n");**

**switch(**option**){**

**case** 1**:**

start **=** create\_and\_add**(**start**);**

**break;**

**case** 2**:**

start **=** display\_all**(**start**);**

**break;**

**case** 3**:**

start **=** search**(**start**);**

**break;**

**case** 4**:**

start **=** delete\_head**(**start**);**

**break;**

**case** 5**:**

start **=** delete\_end**(**start**);**

**break;**

**case** 6**:**

start **=** delete\_search**(**start**);**

**break;**

**case** 7**:**

start **=** delete\_before**(**start**);**

**break;**

**case** 8**:**

start **=** delete\_after**(**start**);**

**break;**

**case** 9**:**

start **=** delete\_except**(**start**);**

**break;**

**case** 10**:**

start **=** delete\_all**(**start**);**

**break;**

**case** 11**:**

start **=** count**(**start**);**

**break;**

**case** 12**:**

start **=** add\_head**(**start**);**

**break;**

**case** 13**:**

start **=** add\_before**(**start**);**

**break;**

**case** 14**:**

start **=** add\_after**(**start**);**

**break;**

**case** 15**:**

start **=** add\_in\_search**(**start**);**

**break;**

**case** 16**:**

start **=** sort\_ascend**(**start**);**

**break;**

**case** 17**:**

start **=** sort\_descen**(**start**);**

**break;**

**case** 18**:**

start **=** insert\_end**(**start**);**

**break;**

**case** 19**:**

start **=** add\_in\_middle**(**start**);**

**break;**

**case** 20**:**

start **=** delete\_the\_middle**(**start**);**

**break;**

**case** **-**1**:**

**break;**

**default:**

printf**("Enter a valid option and try again.\n");**

**}**

printf**("\n");**

**}**

printf**("This program was made with love by Hari Krishna Shah.\nPlease drop a review or a feedback.\n");**

**}**

**struct** node get\_details**(){**

**struct** node details**;**

**float** temp\_total **=** 0**;**

printf**("Enter the name: ");**

scanf**("%s",** **&**details**.**name**);**

printf**("Enter the roll number : ");**

scanf**("%d",** **&**details**.**roll**);**

printf**("Enter the address: ");**

scanf**("%s",** **&**details**.**address**);**

printf**("Enter the course: ");**

scanf**("%s",** **&**details**.**course**);**

printf**("Enter the birth date in the following order\n");**

printf**("\tEnter the day for birth date: ");**

scanf**("%d",** **&**details**.**dob**.**day**);**

printf**("\tEnter the month for birth date: ");**

scanf**("%d",** **&**details**.**dob**.**month**);**

printf**("\tEnter the year for birth date: ");**

scanf**("%d",** **&**details**.**dob**.**year**);**

printf**("Enter the marks in five subjects: ");**

**for(int** i **=** 0**;** i**<**5**;** i**++){**

scanf**("%f",** **&**details**.**marks**[**i**]);**

temp\_total **+=** details**.**marks**[**i**];**

**}**

details**.**total **=** temp\_total**;**

details**.**average **=** temp\_total**/**5**;**

**if(**details**.**average **>=** 90**){**

strcpy**(**details**.**grade**,** **"A+");**

**}**

**else** **if(**details**.**average **>=** 80**){**

strcpy**(**details**.**grade**,** **"A");**

**}**

**else** **if(**details**.**average **>=** 70**){**

strcpy**(**details**.**grade**,** **"B+");**

**}**

**else** **if(**details**.**average **>=** 60**){**

strcpy**(**details**.**grade**,** **"B");**

**}**

**else** **if(**details**.**average **>=** 50**){**

strcpy**(**details**.**grade**,** **"C+");**

**}**

**else** **if(**details**.**average **>=** 40**){**

strcpy**(**details**.**grade**,** **"C");**

**}**

**else{**

strcpy**(**details**.**grade**,** **"F");**

**}**

printf**("The details saved successfully for this node.\n\n");**

**return** details**;**

**}**

**void** display**(struct** node **\***ptr**){**

printf**("The details are given below.\n");**

printf**("Name: %s\n",** ptr**->**name**);**

printf**("Roll Number: %d\n",** ptr**->**roll**);**

printf**("Address: %s\n",** ptr**->**address**);**

printf**("Course: %s\n",** ptr**->**course**);**

printf**("Date of Birth in format day/month/year : %d/%d/%d.\n",** ptr**->**dob**.**day**,** ptr**->**dob**.**month**,**ptr**->**dob**.**year**);**

printf**("Marks Details are below.\n");**

**for(int** i **=** 0**;** i**<**5**;** i**++){**

printf**("Marks for subject %d: %.2f\n",** i**+**1**,** ptr**->**marks**[**i**]);**

**}**

printf**("Total: %.2f\n",** ptr**->**total**);**

printf**("Average: %.2f\n",** ptr**->**average**);**

printf**("Grade: %s\n",** ptr**->**grade**);**

**}**

**struct** node **\***create\_and\_add**(struct** node **\***start**){**

**int** node\_count **=** 0**;**

**struct** node **\***new\_node**,** **\***ptr**;**

ptr **=** start**;**

printf**("Enter the number of nodes you want to add: ");**

scanf**("%d",** **&**node\_count**);**

**if(**start **==** NULL**){**

new\_node **=** **(struct** node **\*)** **(**malloc**(sizeof(struct** node**)));**

new\_node**->**previous **=** NULL**;**

new\_node**->**next **=** NULL**;**

printf**("Enter the details of student below.\n");**

**\***new\_node **=** get\_details**();**

node\_count **=** node\_count **-** 1**;**

start **=** new\_node**;**

**}**

**for(int** i **=** 0**;** i**<**node\_count**;** i**++){**

ptr **=** start**;**

**while(**ptr**->**next **!=** NULL**){**

ptr **=** ptr**->**next**;**

**}**

new\_node **=** **(struct** node **\*)** **(**malloc**(sizeof(struct** node**)));**

printf**("Enter the details of student below.\n");**

**\***new\_node **=** get\_details**();**

new\_node**->**previous **=** ptr**;**

new\_node**->**next **=** NULL**;**

ptr**->**next **=** new\_node**;**

**}**

printf**("All the data saved successfully.\n");**

**return** start**;**

**}**

**struct** node **\***display\_all**(struct** node **\***start**){**

**struct** node **\***ptr**;**

ptr **=** start**;**

**int** count **=** 0**;**

**if(**ptr **==** NULL**){**

printf**("The linked list is empty. Add some nodes first and try again.\n");**

**return** start**;**

**}**

printf**("All the student's detail are given below.\n\n");**

**while(**ptr **!=** NULL**){**

printf**("For student number %d:\n",** count**+**1**);**

display**(**ptr**);**

printf**("\n");**

ptr **=** ptr**->**next**;**

count**+=**1**;**

**}**

**return** start**;**

**}**

**struct** node **\***search**(struct** node **\***start**){**

**struct** node **\***ptr**,** **\***match**,** **\***succedptr**,** **\***preptr**;**

ptr **=** start**;**

**if(**start **==** NULL**){**

printf**("The linked list is empty. Add some nodes first and try again.\n");**

**return** start**;**

**}**

**int** search**;**

printf**("Enter a roll number to search in the linked list: ");**

scanf**("%d",** **&**search**);**

printf**("\n");**

**while(**ptr**->**roll **!=** search**){**

ptr **=** ptr**->**next**;**

**if(**ptr **==** NULL**){**

printf**("The searched roll number doesn't exist in the linked list.\n");**

**return** start**;**

**}**

**}**

match **=** ptr**;**

succedptr **=** match**->**next**;**

preptr **=** match**->**previous**;**

printf**("\n\nHere is the record of the searched student.\n");**

display**(**match**);**

**if(**match **==** start**){**

printf**("\nThe searched roll number is the first student in the list.\nSo, the detail of student before it can't be displayed.\n");**

**}**

**else** **if(**match **!=** start**){**

printf**("\nThe detail of previous student of searched student is given below.\n");**

display**(**preptr**);**

printf**("\n");**

**}**

**if(**start**->**next **==** NULL**){**

printf**("\nThe searched roll number is the last student in the list.\nSo, the detail of student after it can't be displayed.\n");**

**}**

**else** **if(**match**->**next **!=** NULL**){**

printf**("\nHere is the record of student after the searched student\n");**

display**(**succedptr**);**

printf**("\n");**

**}**

printf**("\n");**

**return** start**;**

**}**

**struct** node **\***delete\_head**(struct** node **\***start**){**

**struct** node **\***temp1**,** **\***temp2**;**

temp2 **=** start**;**

**if(**start **==** NULL**){**

printf**("The linked list is empty.\n\n");**

**return** start**;**

**}**

**if(**start**->**next **==** NULL**){**

start **=** NULL**;**

printf**("The linked list contains only one element.\nSo,the first node is also the last node.\nThe first node deleted successfuly.\n");**

free**(**temp2**);**

**return** start**;**

**}**

temp1 **=** start**->**next**;**

temp1**->**previous **=** NULL**;**

free**(**temp2**);**

start **=** temp1**;**

printf**("The first node has been deleted sucessfully.\n\n");**

**return** start**;**

**}**

**struct** node **\***delete\_end**(struct** node **\***start**){**

**if(**start **==** NULL**){**

printf**("The linked list is empty.\n\n");**

**return** start**;**

**}**

**if(**start**->**next **==** NULL**){**

printf**("Currenlty, there is only one node in the linked list so, the first node is also the last node.\n");**

start **=** delete\_head**(**start**);**

**return** start**;**

**}**

**struct** node **\***ptr**,** **\***temp**;**

ptr **=** start**;**

**while(**ptr**->**next **!=** NULL**){**

ptr **=** ptr**->**next**;**

**}**

temp **=** ptr**;**

ptr **=** ptr**->**previous**;**

ptr**->**next **=** NULL**;**

free**(**temp**);**

printf**("The last node has been deleted sucessfully.\n\n");**

**return** start**;**

**}**

**struct** node **\***delete\_search**(struct** node **\***start**){**

**struct** node **\***ptr**,** **\***preptr**;**

**int** search**;**

**if(**start **==** NULL**){**

printf**("The linked list is empty. Add some nodes first and try again.\n");**

**return** start**;**

**}**

printf**("Enter the roll number of the student whose record is to be deleted: ");**

scanf**("%d",** **&**search**);**

ptr **=** start**;**

**if(**start**->**roll **==** search**){**

start **=** delete\_head**(**start**);**

**return** start**;**

**}**

ptr **=** start**;**

**while(**ptr**->**roll **!=** search**){**

ptr **=** ptr**->**next**;**

**if(**ptr **==** NULL**){**

printf**("The searched data was not found in the linked list. The process aborted.\n");**

**return** start**;**

**}**

**}**

**if(**ptr**->**next **==** NULL**){**

printf**("The found data is in the last node so last node will be deleted.\n");**

start **=** delete\_end**(**start**);**

**return** start**;**

**}**

preptr **=** ptr**->**previous**;**

ptr **=** ptr**->**next**;**

preptr**->**next **=** ptr**;**

ptr**->**previous **=** preptr**;**

printf**("The searched student's record deleted successfully.\n");**

**return** start**;**

**}**

**struct** node **\***delete\_before**(struct** node **\***start**){**

**struct** node **\***ptr**,** **\***succedptr**,** **\***match**;**

**int** search**;**

**if(**start **==** NULL**){**

printf**("The linked list is empty. Add some nodes first and try again.\n");**

**return** start**;**

**}**

printf**("Enter the roll number of the student whose record is to be deleted: ");**

scanf**("%d",** **&**search**);**

ptr **=** start**;**

**while(**ptr**->**roll **!=** search**){**

ptr **=** ptr**->**next**;**

**if(**ptr **==** NULL**){**

printf**("The searched data was not found in the linked list. The process aborted.\n");**

**return** start**;**

**}**

**}**

**if(**ptr **==** start**){**

printf**("The searched data is in the first node so, the node before it can't be deleted as it doesn't exist.\n");**

**return** start**;**

**}**

**if(**ptr**->**previous **==** start**){**

start **=** delete\_head**(**start**);**

**return** start**;**

**}**

match **=** ptr**;**

succedptr **=** ptr**->**next**;**

ptr **=** ptr**->**previous**;**

ptr**->**next **=** succedptr**;**

succedptr**->**previous **=** ptr**;**

free**(**match**);**

printf**("The student's record which is previous of searched student has been deleted successfully.\n");**

**return** start**;**

**}**

**struct** node **\***delete\_after**(struct** node **\***start**){**

**struct** node **\***ptr**,** **\***match**,** **\***succedptr**;**

**int** search**;**

**if(**start **==** NULL**){**

printf**("The linked list is empty. Add some nodes first and try again.\n");**

**return** start**;**

**}**

printf**("Enter the roll number of the student whose record is to be deleted: ");**

scanf**("%d",** **&**search**);**

ptr **=** start**;**

**while(**ptr**->**roll **!=** search**){**

ptr **=** ptr**->**next**;**

**if(**ptr **==** NULL**){**

printf**("The searched data was not found in the linked list. The process aborted.\n");**

**return** start**;**

**}**

**}**

**if(**ptr**->**next **==** NULL**){**

printf**("The searched data was found to be in the last node of the linked list.\nSo, the node after it can't be deleted.\n");**

**return** start**;**

**}**

match **=** ptr**;**

succedptr **=** ptr**->**next**;**

**if(**succedptr**->**next **==** NULL**){**

start **=** delete\_end**(**start**);**

**return** start**;**

**}**

ptr **=** succedptr**->**next**;**

ptr**->**previous **=** match**;**

match**->**next **=** ptr**;**

free**(**succedptr**);**

printf**("The student's record which is after the searched student has been deleted successfully.\n");**

**return** start**;**

**}**

**struct** node **\***delete\_all**(struct** node **\***start**){**

**struct** node **\***ptr**,** **\***temp**;**

ptr **=** start**;**

**if(**start **==** NULL**){**

printf**("The linked list is empty. Add some nodes first.\n");**

**return** start**;**

**}**

**while(**ptr **!=** NULL**){**

temp **=** ptr**->**next**;**

free**(**ptr**);**

ptr **=** temp**;**

**}**

printf**("The entire student record has been deleted successfully.\n");**

start **=** NULL**;**

**return** start**;**

**}**

**struct** node **\***delete\_except**(struct** node **\***start**){**

**struct** node **\***ptr**;**

**if(**start **==** NULL**){**

printf**("The linked list is empty. Add some nodes first.\n");**

**return** start**;**

**}**

**struct** node **\***temp**;**

temp **=** **(struct** node **\*)** **(**malloc**(sizeof(struct** node**)));**

ptr **=** start**;**

**int** search**;**

printf**("Enter the roll number whose record is to be kept and all other is to be deleted: ");**

scanf**("%d",** **&**search**);**

**while(**ptr**->**roll **!=** search**){**

ptr **=** ptr**->**next**;**

**if(**ptr **==** NULL**){**

printf**("The searched data wasn't found in the linked list. The process aborted.\n");**

**return** start**;**

**}**

**}**

**\***temp **=** **\***ptr**;**

start **=** delete\_all**(**start**);**

start **=** **(struct** node **\*)** **(**malloc**(sizeof(struct** node**)));**

**\***start **=** **\***temp**;**

start**->**next **=** NULL**;**

start**->**previous **=** NULL**;**

**return** start**;**

**}**

**struct** node **\***count**(struct** node **\***start**){**

**struct** node **\***ptr**;**

ptr **=** start**;**

**int** count **=** 0**;**

**while(**ptr **!=** NULL**){**

count **+=** 1**;**

ptr **=** ptr**->**next**;**

**}**

printf**("The number of nodes are %d.\n\n",** count**);**

**return** start**;**

**}**

**struct** node **\***add\_head**(struct** node **\***start**){**

**struct** node **\***new\_node**,** **\***temp **,** **\***temp\_details**;;**

**int** num**;**

temp **=** start**;**

new\_node **=** **(struct** node **\*)** **(**malloc**(sizeof(struct** node**)));**

printf**("Enter the data for structure to be inserted in the first node\n");**

**\***new\_node **=** get\_details**();**

**if(**start **==** NULL**){**

start **=** new\_node**;**

start**->**next **=** NULL**;**

start**->**previous **=** NULL**;**

**}**

**else{**

new\_node**->**next **=** temp**;**

new\_node**->**previous **=** NULL**;**

start **=** new\_node**;**

**}**

printf**("The data has been added to the head of the linked list.\n\n");**

**return** start**;**

**}**

**struct** node **\***add\_after**(struct** node **\***start**){**

**struct** node **\***new\_node**,** **\***ptr**,** **\***temp**;**

**if(**start **==** NULL**){**

printf**("The link is empty. Add some nodes to the list first.\n\n");**

**return** start**;**

**}**

new\_node **=** **(struct** node **\*)** **(**malloc**(sizeof(struct** node**)));**

**int** num**,** check\_data**;**

printf**("Enter the roll number of the node after which the node is to be added: ");**

scanf**("%d",** **&**check\_data**);**

ptr **=** start**;**

**while(**ptr**->**roll **!=** check\_data**){**

ptr **=** ptr**->**next**;**

**if(**ptr **==** NULL**){**

printf**("The searched data couldn't be found. The process terminated.\n\n");**

**return** start**;**

**}**

**}**

printf**("Enter the data to be stored in the node\n");**

**\***new\_node **=** get\_details**();**

temp **=** ptr**->**next**;**

ptr**->**next **=** new\_node**;**

new\_node**->**next **=** temp**;**

new\_node**->**previous **=** ptr**;**

printf**("The node has been added after the data %d.\n\n",** check\_data**);**

**return** start**;**

**}**

**struct** node **\***add\_before**(struct** node **\***start**){**

**struct** node **\***new\_node**,** **\***ptr**,** **\***temp**;**

**if(**start **==** NULL**){**

printf**("The link is empty. Add some nodes to the list first.\n\n");**

**return** start**;**

**}**

**int** num**,** check\_data**;**

printf**("Enter the roll number before which node is to be added: ");**

scanf**("%d",** **&**check\_data**);**

ptr **=** start**;**

**if(**start**->**roll **==** check\_data**){**

printf**("The matched data was found to be in the first node of data.\n");**

printf**("Now we need to add a new head to our linked list.\n");**

start **=** add\_head**(**start**);**

**return** start**;**

**}**

**while(**ptr**->**roll **!=** check\_data**){**

ptr **=** ptr**->**next**;**

**if(**ptr **==** NULL**){**

printf**("The searched data couldn't be found. The process terminated.\n\n");**

**return** start**;**

**}**

**}**

new\_node **=** **(struct** node **\*)** **(**malloc**(sizeof(struct** node**)));**

printf**("Enter the data to be stored in the new node store.\n");**

**\***new\_node **=** get\_details**();**

temp **=** ptr**;**

ptr **=** ptr**->**previous**;**

ptr**->**next **=** new\_node**;**

new\_node**->**previous **=** ptr**;**

new\_node**->**next **=** temp**;**

temp**->**previous **=** new\_node**;**

printf**("The node has been added before the data %d.\n\n",** check\_data**);**

**return** start**;**

**}**

**struct** node **\***add\_in\_search**(struct** node **\***start**){**

**if(**start **==** NULL**){**

printf**("The link is empty. Add some nodes to the list first.\n\n");**

**return** start**;**

**}**

**struct** node **\***ptr**,** **\***succedptr**,** **\***preptr**;**

ptr **=** start**;**

**int** search**;**

printf**("Enter the roll number to search: ");**

scanf**("%d",** **&**search**);**

**while(**ptr**->**roll **!=** search**){**

ptr **=** ptr**->**next**;**

**if(**ptr **==** NULL**){**

printf**("The searched data couldn't be found. The process terminated.\n\n");**

**return** start**;**

**}**

**}**

succedptr **=** ptr**->**next**;**

preptr **=** ptr**->**previous**;**

**\***ptr **=** get\_details**();**

ptr**->**next **=** succedptr**;**

ptr**->**previous **=** preptr**;**

printf**("The new data has been added to the searched roll number successfully.\n");**

**return** start**;**

**}**

**struct** node **\***sort\_ascend**(struct** node **\***start**){**

**if(**start **==** NULL**){**

printf**("The linked list has no nodes. Add some data and try again.\n");**

**return** start**;**

**}**

**if(**start**->**next **==** NULL**){**

printf**("The linked list has only one node so it can't be sorted.\n");**

**return** start**;**

**}**

**struct** node **\***ptr**,** **\***succedptr**;**

**struct** node ptrtemp**,** succtemp**;**

ptr **=** start**;**

**while(**ptr **!=** NULL**){**

succedptr **=** ptr**->**next**;**

**while(**succedptr **!=** NULL**){**

ptrtemp **=** **\***ptr**;**

succtemp **=** **\***succedptr**;**

**if(**ptr**->**roll **>** succedptr**->**roll**){**

**\***succedptr **=** **\***ptr**;**

succedptr**->**next **=** succtemp**.**next**;**

succedptr**->**previous **=** succtemp**.**previous**;**

succtemp**.**next **=** ptrtemp**.**next**;**

succtemp**.**previous **=** ptrtemp**.**previous**;**

**\***ptr **=** succtemp**;**

**}**

succedptr **=** succedptr**->**next**;**

**}**

ptr **=** ptr**->**next**;**

**}**

printf**("The linked list has been sorted in ascending order successfully.\n");**

**return** start**;**

**}**

**struct** node **\***sort\_descen**(struct** node **\***start**){**

**struct** node **\***ptr**,** **\***succedptr**;**

**struct** node ptrtemp**,** succtemp**;**

ptr **=** start**;**

**if(**start **==** NULL**){**

printf**("The linked list has no nodes. Add some data and try again.\n");**

**return** start**;**

**}**

**if(**start**->**next **==** NULL**){**

printf**("The linked list has only one node so it can't be sorted.\n");**

**return** start**;**

**}**

**while(**ptr **!=** NULL**){**

succedptr **=** ptr**->**next**;**

**while(**succedptr **!=** NULL**){**

ptrtemp **=** **\***ptr**;**

succtemp **=** **\***succedptr**;**

**if(**ptr**->**roll **<** succedptr**->**roll**){**

**\***succedptr **=** **\***ptr**;**

succedptr**->**next **=** succtemp**.**next**;**

succedptr**->**previous **=** succtemp**.**previous**;**

succtemp**.**next **=** ptrtemp**.**next**;**

succtemp**.**previous **=** ptrtemp**.**previous**;**

**\***ptr **=** succtemp**;**

**}**

succedptr **=** succedptr**->**next**;**

**}**

ptr **=** ptr**->**next**;**

**}**

printf**("The linked list has been sorted in descending order successfully.\n");**

**return** start**;**

**}**

**struct** node **\***insert\_end**(struct** node **\***start**){**

**if(**start **==** NULL**){**

printf**("The linked list is empty.\nAdd some nodes first.\n");**

start **=** add\_head**(**start**);**

**return** start**;**

**}**

**struct** node **\***ptr**,** **\***new\_node**;**

new\_node **=** **(struct** node **\*)** **(**malloc**(sizeof(struct** node**)));**

ptr **=** start**;**

**while(**ptr**->**next **!=** NULL**){**

ptr **=** ptr**->**next**;**

**}**

**\***new\_node **=** get\_details**();**

new\_node**->**next**=**NULL**;**

new\_node**->**previous **=** ptr**;**

ptr**->**next **=** new\_node**;**

printf**("The node has been added at the end.\n\n");**

**return** start**;**

**}**

**struct** node **\***add\_in\_middle**(struct** node **\***start**){**

**if(**start **==** NULL**){**

printf**("The linked list is empty. Add some nodes first.\n\n");**

**return** start**;**

**}**

**struct** node **\***ptr**,** **\***new\_node**,** **\***temp**;**

new\_node **=** **(struct** node **\*)** **(**malloc**(sizeof(struct** node**)));**

**int** count **=** 0**,** middle**;**

ptr **=** start**;**

**while(**ptr **!=** NULL**){**

ptr **=** ptr**->**next**;**

count **+=** 1**;**

**}**

**if(**count **%** 2 **==** 0**){**

middle **=** **(**count**/**2**)-**1**;**

**}**

**else{**

middle **=** **((**count**-**1**)/**2**);**

**}**

ptr **=** start**;**

**for(int** i **=** 0**;** i**<**middle**;** i**++){**

ptr **=** ptr**->**next**;**

**}**

printf**("Enter the details of the student to add at the middle node below.\n");**

**\***new\_node **=** get\_details**();**

**if(**start**->**next **==** NULL**){**

start**->**next **=** new\_node**;**

new\_node**->**next **=** NULL**;**

new\_node**->**previous **=** start**;**

**return** start**;**

**}**

temp **=** ptr**->**next**;**

new\_node**->**next **=** ptr**->**next**;**

new\_node**->**previous **=** ptr**;**

ptr**->**next **=** new\_node**;**

temp**->**previous **=** new\_node**;**

printf**("The middle node has been added succesfully.\n");**

**return** start**;**

**}**

**struct** node **\***delete\_the\_middle**(struct** node **\***start**){**

**if(**start **==** NULL**){**

printf**("The linked list is empty. Add some nodes first.\n\n");**

**return** start**;**

**}**

**if(**start**->**next **==** NULL**){**

printf**("The linked list has only 1 node so it's middle node can't be deleted.\n");**

**return** start**;**

**}**

**struct** node **\***ptr**,** **\***succedptr**,** **\***mid\_node**;**

**int** count **=** 0**,** middle**;**

ptr **=** start**;**

**while(**ptr **!=** NULL**){**

ptr **=** ptr**->**next**;**

count **+=** 1**;**

**}**

**if(**count **%** 2 **==** 0**){**

middle **=** **(**count**/**2**);**

**}**

**else{**

middle **=** **((**count**-**1**)/**2**);**

**}**

ptr **=** start**;**

**for(int** i **=** 0**;** i**<**middle**;** i**++){**

ptr **=** ptr**->**next**;**

**}**

mid\_node **=** ptr**;**

**if(**mid\_node**->**next **==** NULL**){**

ptr **=** mid\_node**->**previous**;**

ptr**->**next **=** NULL**;**

free**(**mid\_node**);**

**return** start**;**

**}**

succedptr **=** ptr**->**next**;**

ptr **=** ptr**->**previous**;**

ptr**->**next **=** succedptr**;**

succedptr**->**previous **=** ptr**;**

free**(**mid\_node**);**

printf**("The middle node has been deleted succesfully.\n");**

**return** start**;**

**}**

