***6.*** ***WAP to Implement Singly Linked List with following operations a) a) Create a linked list. b) Deletion of first element, specified element and last element in the list. c) Display the contents of the linked list***.

***#include <stdlib.h>***

***#include <stdio.h>***

***void create();***

***void display();***

***void insert\_begin();***

***void insert\_end();***

***void insert\_pos();***

***struct node***

***{***

***int info;***

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

***};***

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

***int main()***

***{***

***int choice;***

***while (1)***

***{***

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

***printf("\n 1.Create \n");***

***printf("\n 2.Display \n");***

***printf("\n 3.Insert at the beginning \n");***

***printf("\n 4.Insert at the end \n");***

***printf("\n 5.Insert at specified position \n");***

***printf("\n 6.Exit \n");***

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

***printf("Enter your choice:");***

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

***switch (choice)***

***{***

***case 1:***

***create();***

***break;***

***case 2:***

***display();***

***break;***

***case 3:***

***insert\_begin();***

***break;***

***case 4:***

***insert\_end();***

***break;***

***case 5:***

***insert\_pos();***

***break;***

***case 6:***

***exit(0);***

***break;***

***default:***

***printf("\n Wrong Choice:\n");***

***break;***

***}***

***}***

***return 0;***

***}***

***void create()***

***{***

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

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

***if (temp == NULL)***

***{***

***printf("\nOut of Memory Space:\n");***

***exit(0);***

***}***

***printf("\nEnter the data value for the node:");***

***scanf("%d", &temp->info);***

***temp->next = NULL;***

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

***{***

***start = temp;***

***}***

***else***

***{***

***ptr = start;***

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

***{***

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

***}***

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

***}***

***}***

***void display()***

***{***

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

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

***{***

***printf("\nList is empty:\n");***

***return;***

***}***

***else***

***{***

***ptr = start;***

***printf("\nThe List elements are:\n");***

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

***{***

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

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

***}***

***}***

***}***

***void insert\_begin()***

***{***

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

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

***if (temp == NULL)***

***{***

***printf("\nOut of Memory Space:\n");***

***return;***

***}***

***printf("\nEnter the data value for the node:");***

***scanf("%d", &temp->info);***

***temp->next = NULL;***

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

***{***

***start = temp;***

***}***

***else***

***{***

***temp->next = start;***

***start = temp;***

***}***

***}***

***void insert\_end()***

***{***

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

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

***if (temp == NULL)***

***{***

***printf("\nOut of Memory Space:\n");***

***return;***

***}***

***printf("\nEnter the data value for the node:");***

***scanf("%d", &temp->info);***

***temp->next = NULL;***

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

***{***

***start = temp;***

***}***

***else***

***{***

***ptr = start;***

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

***{***

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

***}***

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

***}***

***}***

***void insert\_pos()***

***{***

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

***int i, pos;***

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

***if (temp == NULL)***

***{***

***printf("\nOut of Memory Space:\n");***

***return;***

***}***

***printf("\nEnter the position for the new node to be inserted:");***

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

***printf("\nEnter the data value of the node:");***

***scanf("%d", &temp->info);***

***temp->next = NULL;***

***if (pos == 0)***

***{***

***temp->next = start;***

***start = temp;***

***}***

***else***

***{***

***for (i = 0, ptr = start; i < pos - 1; i++)***

***{***

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

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

***{***

***printf("\nPosition not found:\n");***

***return;***

***}***

***}***

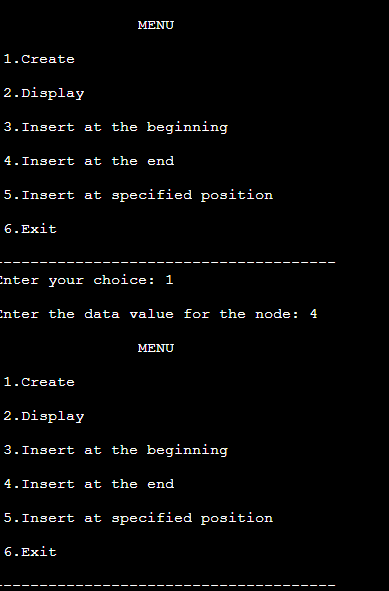
***temp->next = ptr->next;***

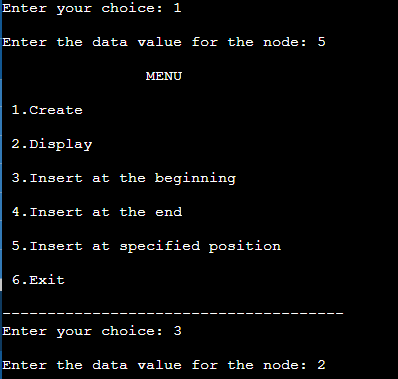
***ptr->next = temp;***

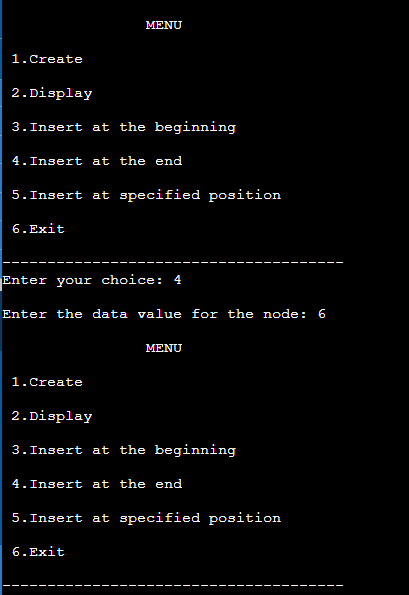
***}***

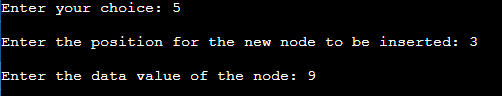
***}***

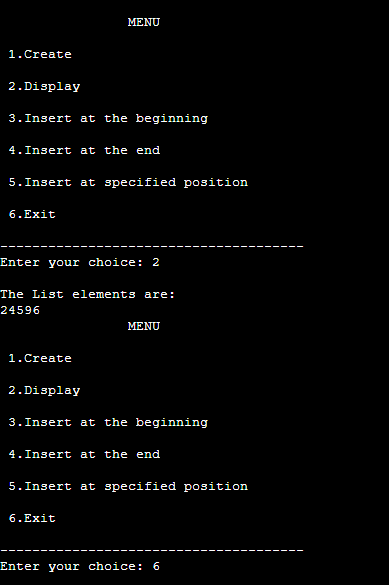
***Output:***

******

******

******

******

******