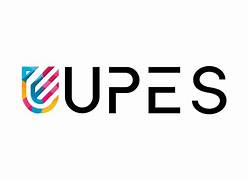
**DSA**



Submitted by,

**Name - Viswas PP**

**SAP ID - 500125032**

**Enrollment ID - R2142231539**

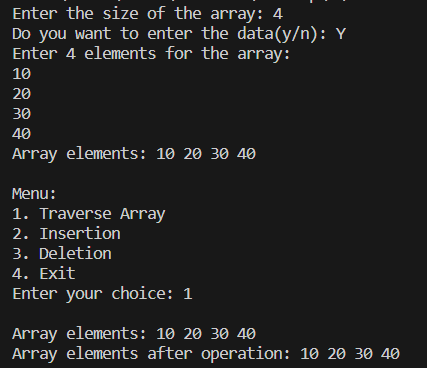
**Batch - 46**

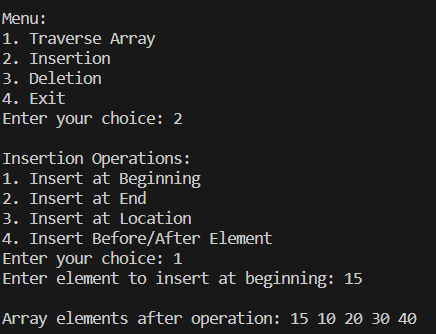
**Signature**

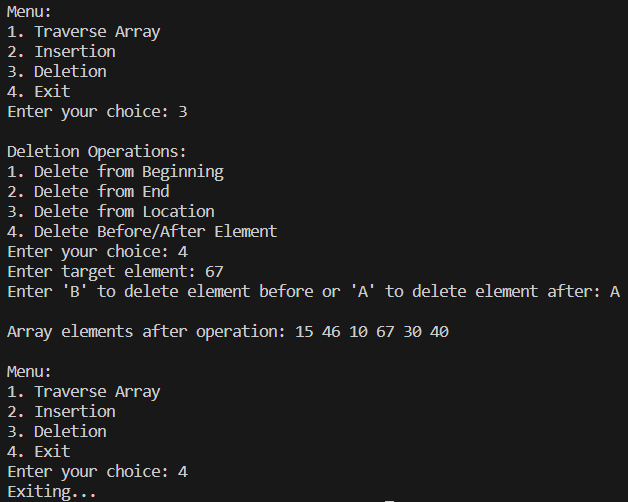
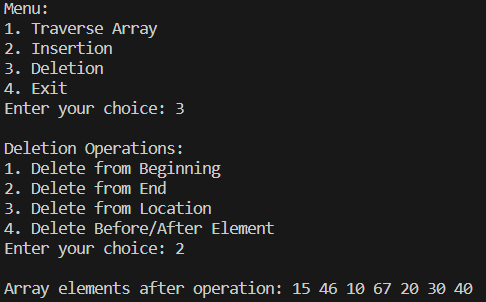
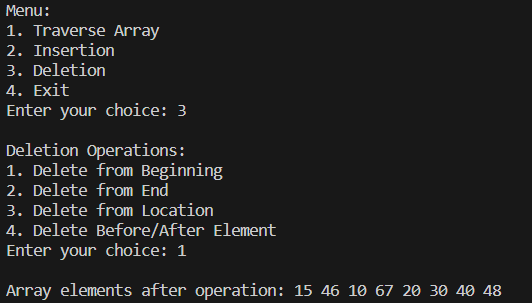
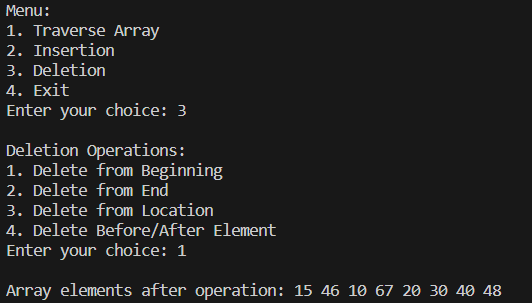
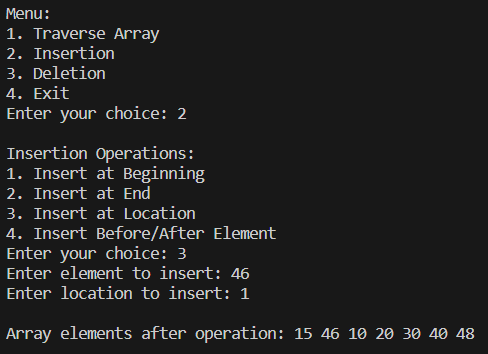
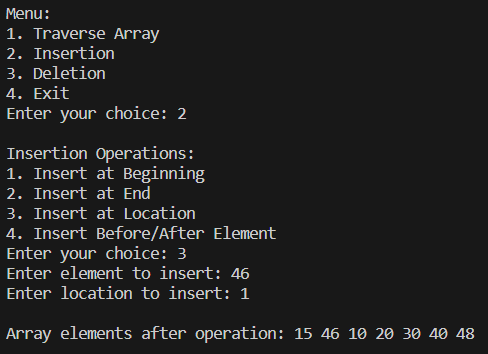
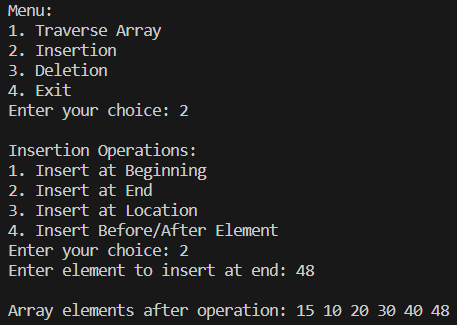
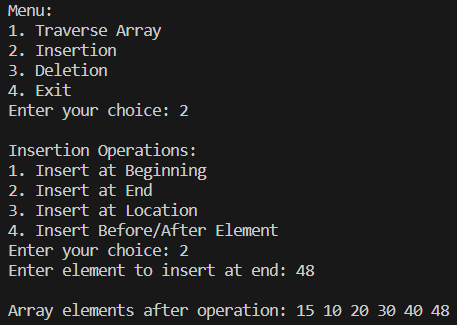
**EXPERIMENT - 1**

**OPERATIONS ON ARRAY**

#include <stdio.h>#include <stdlib.h>void traverse(int arr[], int size);void ins\_Beg(int arr[], int \*size, int element);void ins\_End(int arr[], int \*size, int element);void ins\_Loc(int arr[], int \*size, int element, int location);void ins\_BeforeAfterElem(int arr[], int \*size, int element, int target, char position);void del\_Beg(int arr[], int \*size);void del\_End(int arr[], int \*size);void del\_Loc(int arr[], int \*size, int location);void del\_BeforeAfterElem(int arr[], int \*size, int target, char position);int main() { int choice, size; int \*arr = NULL; char c; printf("Enter the size of the array: "); scanf("%d", &size); // Allocate memory for the array (plus one additional element for insertion) arr = (int \*)malloc((size + 1) \* sizeof(int)); if (arr == NULL) { printf("Memory allocation failed.\n"); return -1; } printf("Do you want to enter the data(y/n): "); scanf(" %c",&c); if(c=='y' || c=='Y') { for (int i = 0; i < size; i++) { arr[i] = 0; } printf("Enter %d elements for the array:\n", size); for (int i = 0; i < size; i++) { scanf("%d", &arr[i]); } printf("Array elements: "); traverse(arr, size); printf("\n"); } else { printf("Array is empty"); } do { printf("\nMenu:\n"); printf("1. Traverse Array\n"); printf("2. Insertion\n"); printf("3. Deletion\n"); printf("4. Exit\n"); printf("Enter your choice: "); scanf("%d", &choice); switch (choice) { case 1: printf("\nArray elements: "); traverse(arr, size); break; case 2: printf("\nInsertion Operations:\n"); printf("1. Insert at Beginning\n2. Insert at End\n3. Insert at Location\n4. Insert Before/After Element\nEnter your choice: "); int insertChoice; scanf("%d", &insertChoice); switch (insertChoice) { case 1: { int element; printf("Enter element to insert at beginning: "); scanf("%d", &element); ins\_Beg(arr, &size, element); traverse(arr,size); break; } case 2: { int element; printf("Enter element to insert at end: "); scanf("%d", &element); ins\_End(arr, &size, element); traverse(arr,size); break; } case 3: { int element, location; printf("Enter element to insert: "); scanf("%d", &element); printf("Enter location to insert: "); scanf("%d", &location); ins\_Loc(arr, &size, element, location); traverse(arr,size); break; } case 4: { int element, target; char position; printf("Enter element to insert: "); scanf("%d", &element); printf("Enter target element: "); scanf("%d", &target); printf("Enter 'B' to insert before or 'A' to insert after: "); scanf(" %c", &position); ins\_BeforeAfterElem(arr, &size, element, target, position); traverse(arr,size); break; } default: printf("Invalid choice!\n"); } break; case 3: printf("\nDeletion Operations:\n"); printf("1. Delete from Beginning\n2. Delete from End\n3. Delete from Location\n4. Delete Before/After Element\nEnter your choice: "); int deleteChoice; scanf("%d", &deleteChoice); switch (deleteChoice) { case 1: del\_Beg(arr, &size); traverse(arr,size); break; case 2: del\_End(arr, &size); traverse(arr,size); break; case 3: { int location; printf("Enter location to delete: "); scanf("%d", &location); del\_Loc(arr, &size, location); traverse(arr,size); break; } case 4: { int target; char position; printf("Enter target element: "); scanf("%d", &target); printf("Enter 'B' to delete element before or 'A' to delete element after: "); scanf(" %c", &position); del\_BeforeAfterElem(arr, &size, target, position); traverse(arr,size); break; } default: printf("Invalid choice!\n"); } break; case 4: printf("Exiting...\n"); break; default: printf("Invalid choice!\n"); } if (choice != 4) { printf("\nArray elements after operation: "); traverse(arr,size); printf("\n"); } } while (choice != 4); free(arr); return 0;}void traverse(int arr[], int size) { for (int i = 0; i < size; i++) printf("%d ", arr[i]);}void ins\_Beg(int arr[], int \*size, int element) { // Shift elements to the right for (int i = \*size - 1; i >= 0; i--) arr[i + 1] = arr[i]; arr[0] = element; (\*size)++;}void ins\_End(int arr[], int \*size, int element) { arr[\*size] = element; (\*size)++;}void ins\_Loc(int arr[], int \*size, int element, int location) { // Shift elements to the right from the given location for (int i = \*size - 1; i >= location; i--) arr[i + 1] = arr[i]; arr[location] = element; (\*size)++;}void ins\_BeforeAfterElem(int arr[], int \*size, int element, int target, char position) { int location; for (int i = 0; i < \*size; i++) { if (arr[i] == target) { location = i; break; } } if (position == 'B') ins\_Loc(arr, size, element, location - 1); else if (position == 'A') ins\_Loc(arr, size, element, location + 1);}void del\_Beg(int arr[], int \*size) { // Shift elements to the left for (int i = 0; i < \*size - 1; i++) arr[i] = arr[i + 1]; (\*size)--;}void del\_End(int arr[], int \*size) { (\*size)--;}void del\_Loc(int arr[], int \*size, int location) { // Shift elements to the left from the given location for (int i = location; i < \*size - 1; i++) arr[i] = arr[i + 1]; (\*size)--;}void del\_BeforeAfterElem(int arr[], int \*size, int target, char position) { int location; for (int i = 0; i < \*size; i++) { if (arr[i] == target) { location = i; break; } } if (position == 'B') del\_Loc(arr, size, location - 1); else if (position == 'A') del\_Loc(arr, size, location + 1);}







**EXPERIMENT – 2**

**SEARCHING IN AN ARRAY**

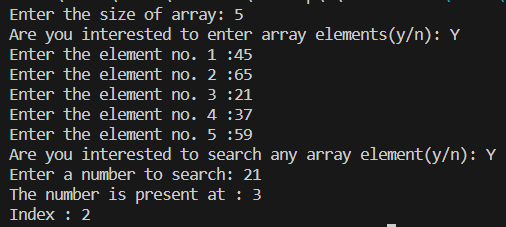
#include<stdio.h>int main() { int limit, i, searchkey; char c; printf("Enter the size of array: "); scanf("%d", &limit); int a[limit]; printf("Are you interested to enter array elements(y/n): "); scanf(" %c",&c); if(c=='y' || c=='Y') { for (i = 0; i < limit; i++) { printf("Enter the element no. %d :",i+1); scanf("%d", &a[i]); } } else { printf("Array is empty\nSearch is not possible!!!"); return 0; } printf("Are you interested to search any array element(y/n): "); scanf(" %c",&c); if(c=='y' || c=='Y') { printf("Enter a number to search: "); scanf("%d", &searchkey); for (i = 0; i < limit; i++) { if (searchkey == a[i]) { printf("The number is present at : %d\nIndex : %d", i + 1,i); break; } } if (i == limit) { printf("The number is not present in the array.");

}

}

return 0;

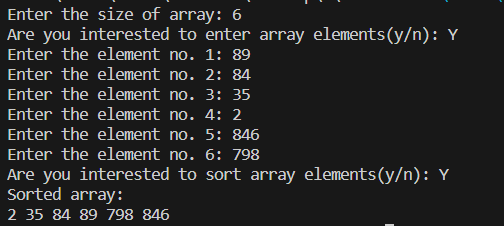
}



**EXPERIMENT – 3**

**SORTING IN AN ARRAY(BUBBLE SORT)**

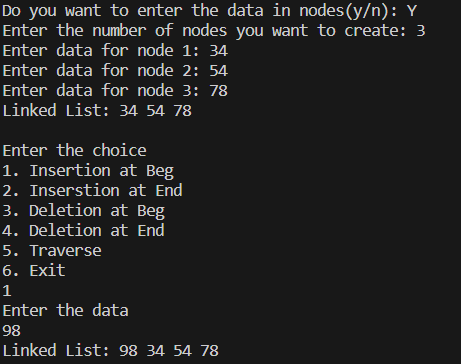
#include <stdio.h>#include <stdlib.h>int main() { int n; char c; printf("Enter the size of array: "); scanf("%d", &n); int arr[n]; printf("Are you interested to enter array elements(y/n): "); scanf(" %c",&c); if(c=='y' || c=='Y') { for (int i = 0; i < n; i++) { printf("Enter the element no. %d: ",i+1); scanf("%d", &arr[i]); } } else { printf("There is no elements\nSorting is impossible"); return 0; } printf("Are you interested to sort array elements(y/n): "); scanf(" %c",&c); if(c=='y' || c=='Y') { for (int i = 0; i < n-1; i++) { for (int j = 0; j < n-i-1; j++) { if (arr[j] > arr[j+1]) { int temp = arr[j]; arr[j] = arr[j+1]; arr[j+1] = temp; } } } printf("Sorted array: \n"); for (int i = 0; i < n; i++) { printf("%d ", arr[i]); } printf("\n"); } return 0;}

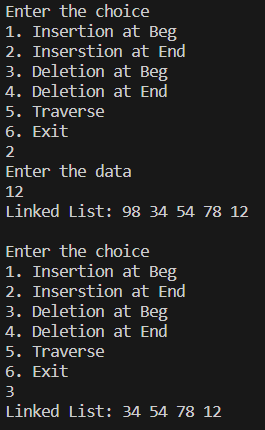
****

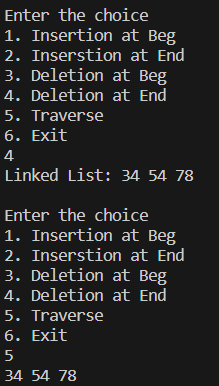
**EXPERIMENT – 4**

**OPERATIONS ON LINKED LIST**

**#include<stdio.h>#include<stdlib.h>#include<malloc.h>struct node{ int data; struct node \*next;};struct node \*head = NULL;struct node \*temp = NULL;void ins\_beg(struct node\*);void ins\_end(struct node\*);void del\_beg();void del\_end();void traverse();void print\_list(); void create\_nodes(); int main(){ int choice, n; struct node \*new1; char c; printf("Do you want to enter the data in nodes(y/n): "); scanf(" %c",&c); if(c=='y' || c=='Y') { create\_nodes(); print\_list(); } else { printf("!Empty Node!"); } while(1) { printf("\nEnter the choice\n"); printf("1. Insertion at Beg\n2. Inserstion at End\n3. Deletion at Beg\n4. Deletion at End\n5. Traverse\n6. Exit\n"); scanf("%d",&choice); switch(choice){ case 1: new1 = (struct node\*)malloc(sizeof(struct node)); printf("Enter the data\n"); scanf("%d",&new1->data); new1->next = NULL; ins\_beg(new1); print\_list(); break; case 2: new1 = (struct node\*)malloc(sizeof(struct node)); printf("Enter the data\n"); scanf("%d",&new1->data); new1->next = NULL; ins\_end(new1); print\_list(); break; case 3: del\_beg(); print\_list(); break; case 4: del\_end(); print\_list(); break; case 5: traverse(); break; default: exit(1); } }}void ins\_beg(struct node \*new1){ if(head == NULL) head = new1; else { new1->next = head; head = new1; }}void ins\_end(struct node \*new1){ temp = head; if(head == NULL) head = new1; else { while(temp->next!= NULL) temp = temp->next; temp->next = new1; }}void del\_beg(){ struct node \*temp = head; if(head == NULL) printf("No item to delete\n"); else head = head->next; free(temp); temp = NULL;}void del\_end(){ struct node \*temp = head; struct node \*temp1 = head; if(head == NULL) printf("No item to delete\n"); else { while(temp1->next != NULL) { temp = temp1; temp1 = temp1->next; } if(temp1 == head) head = NULL; else temp->next = NULL; } free(temp1); temp1 = NULL;}void traverse(){ struct node \*temp = head; if(head == NULL) printf("No item to traverse\n"); while(temp != NULL) { printf("%d ",temp->data); temp = temp->next; }}void print\_list() { printf("Linked List: "); traverse(); printf("\n");}void create\_nodes() { int num\_nodes, data; struct node \*new\_node; printf("Enter the number of nodes you want to create: "); scanf("%d", &num\_nodes); for(int i = 0; i < num\_nodes; i++) { new\_node = (struct node\*)malloc(sizeof(struct node)); printf("Enter data for node %d: ", i+1); scanf("%d", &data); new\_node->data = data; new\_node->next = NULL; if(head == NULL) head = new\_node; else { struct node \*temp = head; while(temp->next != NULL) temp = temp->next; temp->next = new\_node; } }}**

****

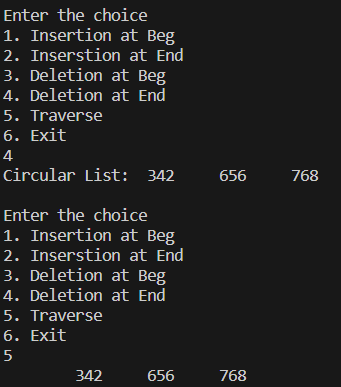
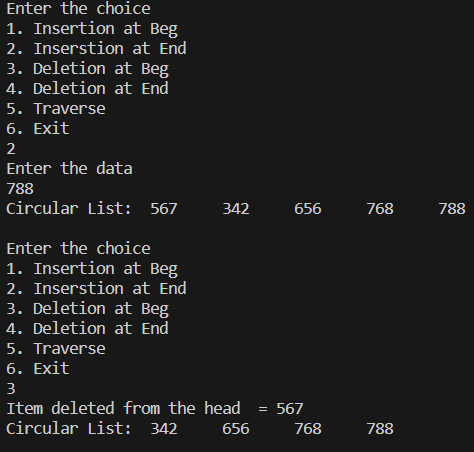
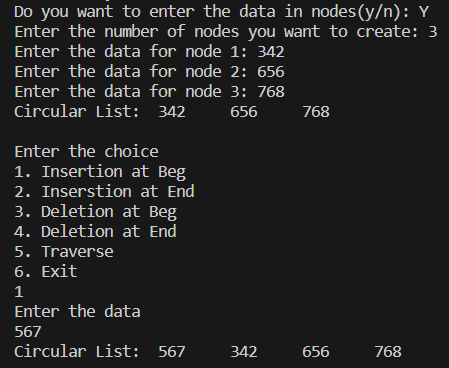




**EXPERIMENT – 5**

**OPERATIONS ON CIRCULAR LINKED LIST**

#include<stdio.h>#include<stdlib.h>#include<malloc.h>struct node{ int data; struct node \*next;};struct node \*head = NULL;void ins\_beg(struct node\*);void ins\_end(struct node\*);void del\_beg();void del\_end();void traverse();void print\_list();void create\_nodes();int main(){ int choice, n; struct node \*new1; char c; printf("Do you want to enter the data in nodes(y/n): "); scanf(" %c",&c); if(c=='y' || c=='Y') { create\_nodes(); print\_list(); } else { printf("!Empty Node!"); } while(1) { printf("\nEnter the choice\n"); printf("1. Insertion at Beg\n2. Inserstion at End\n3. Deletion at Beg\n4. Deletion at End\n5. Traverse\n6. Exit\n"); scanf("%d",&choice); switch(choice){ case 1: new1 = (struct node\*)malloc(sizeof(struct node)); printf("Enter the data\n"); scanf("%d",&new1->data); new1->next = NULL; ins\_beg(new1); print\_list(); break; case 2: new1 = (struct node\*)malloc(sizeof(struct node)); printf("Enter the data\n"); scanf("%d",&new1->data); new1->next = NULL; ins\_end(new1); print\_list(); break; case 3: del\_beg(); print\_list(); break; case 4: del\_end(); print\_list(); break; case 5: traverse(); break; default: exit(1); } }}void ins\_beg(struct node \*new1){ struct node \*temp = head; if(head == NULL) { head = new1; head->next = head; } else{ while(temp->next != head) temp = temp->next; new1->next = head; temp->next = new1; head = new1; }}void ins\_end(struct node \*new1){ struct node \*temp = head; if(head == NULL) { head = new1; head->next = head; } else { while(temp->next!= head) temp = temp->next; temp->next = new1; new1->next = head; }}void del\_beg(){ struct node \*temp = head; struct node \*temp1 = head; if(head == NULL) printf("No elements in the list\n"); else if(head->next == head){ printf("Item deleted from the head = %d\n",head->data); head = NULL; } else { while(temp->next!=head) temp = temp->next; printf("Item deleted from the head = %d\n",head->data); head = head->next; temp->next = head; free(temp1); temp1 = NULL; }}void del\_end(){ struct node \*temp = head; struct node \*temp1 = head; if(head == NULL) { printf("No elements in the list"); } else if(head->next == head) { head = NULL; free(temp); } else{ while(temp1->next != head) temp1 = temp1->next; while(temp->next != temp1) temp = temp->next; temp1->next = NULL; temp->next = head; free(temp1); temp1 = NULL; }}void traverse(){ struct node \*temp = head; if(head == NULL) printf("No elements in the list\n"); else { do { printf("\t%d",temp->data); temp = temp->next; }while(temp!= head); }}void print\_list() { printf("Circular List: "); traverse(); printf("\n");}void create\_nodes() { int num\_nodes; printf("Enter the number of nodes you want to create: "); scanf("%d", &num\_nodes); for (int i = 0; i < num\_nodes; ++i) { struct node \*new\_node = (struct node\*)malloc(sizeof(struct node)); printf("Enter the data for node %d: ", i + 1); scanf("%d", &new\_node->data); new\_node->next = NULL; if (head == NULL) { head = new\_node; head->next = head; } else { ins\_end(new\_node); } }}



**EXPERIMENT – 5**

**OPERATIONS ON DOUBLY LINKED LIST**

#include<stdio.h>#include<stdlib.h>struct node{ int data; struct node \*next; struct node \*prev;};struct node \*head = NULL;//struct node \*temp;void ins\_beg(struct node\*);void ins\_end(struct node\*);void del\_beg();void del\_end();void traverse1();void traverse2();void print\_list();void create\_nodes(int);int main(){ int choice, n; struct node \*new1; char c; printf("Do you want to enter the data in nodes(y/n): "); scanf(" %c",&c); if(c=='y' || c=='Y') { printf("Enter the number of nodes you want to create: "); scanf("%d", &n); create\_nodes(n); print\_list(); } else { printf("!Empty Node!"); } while(1) { printf("\nenter the choice\n"); printf("1. Insertion at Beg\n2. Inserstion at End\n3. Deletion at Beg\n4. Deletion at End\n5. Traverse\n6. Exit\n"); scanf("%d", &choice); switch(choice) { case 1: new1 = (struct node\*)malloc(sizeof(struct node)); printf("Enter the data\n"); scanf("%d", &new1->data); new1->next = NULL; new1->prev = NULL; ins\_beg(new1); print\_list(); break; case 2: new1 = (struct node\*)malloc(sizeof(struct node)); printf("Enter the data\n"); scanf("%d",&new1->data); new1->next = NULL; new1->prev = NULL; ins\_end(new1); print\_list(); break; case 3: del\_beg(); print\_list(); break; case 4: del\_end(); print\_list(); break; case 5: traverse1(); print\_list(); break; default: exit(1); } }} void ins\_beg(struct node \*new1) { if(head == NULL) head = new1; else { head->prev = new1; new1->next = head; head = new1; } } void ins\_end(struct node \*new1) { struct node \*temp = head; if(head == NULL) head = new1; else { while(temp->next!= NULL) temp = temp->next; new1->prev = temp; temp->next = new1; } } void del\_beg() { struct node \*temp = head; if(head == NULL) printf("No item to delete\n"); else head = head->next; head->prev = NULL; free(temp); temp = NULL; } void del\_end() { struct node \*temp = head; struct node \*temp1 = head; if(head == NULL) printf("No item to delete\n"); else { while(temp1->next != NULL) { temp = temp1; temp1 = temp1->next; } if(temp1 == head) head = NULL; else temp->next = NULL; temp1->prev = NULL; free(temp1); temp1 = NULL; } } void traverse1() { struct node \*temp = head; while(temp != NULL) { printf("%d ",temp->data); temp = temp->next; } } void print\_list() { printf("Doubly Linked List: \n"); printf("traverses the list: \n"); traverse1(); printf("\n");} void create\_nodes(int n) { int i; struct node \*temp, \*new\_node; for(i = 0; i < n; i++) { new\_node = (struct node\*)malloc(sizeof(struct node)); printf("Enter data for node %d: ", i+1); scanf("%d", &new\_node->data); new\_node->next = NULL; new\_node->prev = NULL; if(head == NULL) { head = new\_node; } else { temp->next = new\_node; new\_node->prev = temp; } temp = new\_node; }} 