Question 1. Write a C program to design a 5D array, store integer values in it and print them.

Program:

```
#include <stdio.h>
#include <stdlib.h>
int main(){
  printf("Enter the dimensions of the array:\n");
  int n1,n2,n3,n4,n5;
  scanf("%d%d%d%d%d", &n1,&n2,&n3,&n4,&n5);
  int***** arr;
  arr = (int*****)malloc(sizeof(int****)*n1);
  for(int i=0; i<n1; i++){
    arr[i] = (int****)malloc(sizeof(int***)*n2);
    for(int j=0; j<n2; j++){
      arr[i][j] = (int***)malloc(sizeof(int**)*n3);
      for(int k=0; k<n3; k++){
        arr[i][j][k] = (int**)malloc(sizeof(int*)*n4);
        for(int l=0; l<n4; l++){
          arr[i][j][k][l] = (int*)malloc(sizeof(int)*n5);
        }
      }
    }
  }
  printf("Enter the array elements:\n");
  for(int i=0; i<n1; i++){
    for(int j=0; j<n2; j++){
      for(int k=0; k<n3; k++){
        for(int l=0; l<n4; l++){
          for(int m=0; m<n5; m++){
            scanf("%d", &arr[i][j][k][l][m]);
          }
        }
      }
    }
  }
  printf("The array is:\n");
  for(int i=0; i<n1; i++){
    for(int j=0; j<n2; j++){
      for(int k=0; k<n3; k++){
        for(int l=0; l<n4; l++){
          for(int m=0; m<n5; m++){
            printf("%d ", arr[i][j][k][l][m]);
          }
          printf("\n");
        }
      }
    }
  }
  return 0;
}
```

Output:

```
Enter the dimensions of the array:
2 3 1 2 1
Enter the array elements:
2 5 2 45 2 54 12 54 2 45 2 565
The array is:
2
5
2
45
2
54
12
54
2
45
2
565
```

Question 2. Write a C program to design a Singly Linked List and perform Insertion at end, Insertion at Beginning, Deletion at Begin, Deletion at End, Insertion at Index, Deletion at Index, Deletion of entire List and Display Linked List operation on it. You may create a header file of your own to perform the task.

Program:

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

typedef struct node{
  int data;
  struct node* next;
}node;

node* create_node(int data){
  node* newnode = (node*)malloc(sizeof(node));
  newnode->data = data;
  newnode->next = NULL;
}

void print(node* head){
  node* temp = head;
  printf("List is: ");
  while(temp != NULL){
    printf("%d ",temp->data);
    temp = temp->next;
  }
  printf("\n");
}

void insert_at_beginning(node** head,int data){
  node* newnode = create_node(data);
  if((*head) == NULL){ *head = newnode; }
  else{ newnode->next = *head; *head = newnode; }
}
void insert_at_end(node** head,int data){
  node* newnode = create_node(data);
  if((*head) == NULL){ *head = newnode; return; }
  node* temp = *head;
  while(temp->next != NULL) temp = temp->next;
  temp->next = newnode;
}
void insert_at_index(node** head,int data,int idx){
  node* newnode = create_node(data);
  if(idx==0 || (*head)==NULL){
    insert_at_beginning(head,data);
    return;
  }
  int counter = 0; node* temp = *head;
  while(temp->next!=NULL && counter<idx-1){
    temp = temp->next; counter++;
  }
  newnode->next = temp->next;
  temp->next = newnode;
}

void delete_from_beginning(node** head){
  if((*head) == NULL) return;
  node* tmp = *head;
  *head = (*head)->next;
  free(tmp);
}
void delete_from_end(node** head){
  if((*head) == NULL) return;
  node* temp = *head;
  if((*head)->next == NULL){
    free(*head); head=NULL;
    return;
```

```
  }
  while(temp->next->next != NULL) temp = temp->next;
  node* last_node = temp->next;
  temp->next = NULL;
  free(last_node);
}
void delete_from_index(node** head,int idx){
  if(idx==0 || (*head)==NULL){
    delete_from_beginning(head); return;
  }
  int counter = 0;
  node* temp = *head;
  while(temp->next->next!=NULL && counter<idx-1){
    temp = temp->next; counter++;
  }
  node* to_delete=temp->next;
  temp->next=temp->next->next;
  free(to_delete);
}
void clear_list(node** head){
  node* temp = *head;
  while(temp != NULL){
    *head=NULL;
    free(*head);
    (*head) = temp->next;
    temp = temp->next;
  }
}

#include "linked_list.h"
int main(){
  node* head = NULL;
  while (1) {
    printf("Press 1 to add a node, and 2 to delete a node, and 3 to exit.\n");
    printf("Enter your choice: \n");
    int choice;
    scanf("%d", &choice);
    if(choice == 1) {
      printf("Press 1 to add node at beginning, 2 to add node at end, and 3 to add node after a
certain index.\n");
      printf("Enter your choice: \n");
      scanf("%d", &choice);
      int value;
      printf("Enter the value to be added to the list:\n");
      scanf("%d", &value);
      if (choice == 1) {
        insert_at_beginning(&head, value);
      } else if (choice == 2) {
        insert_at_end(&head, value);
      } else if (choice == 3) {
        printf("Enter the index at which you want to enter the value:\n");
        int index;
        scanf("%d", &index);
        insert_at_index(&head, value, index);
      } else {
        printf("Wrong choice entered!\n");
      }
    } else if (choice == 2) {
      printf("Press 1 to delete node from beginning, 2 to delete node from end, and 3 to delete
node from any index.\n");
      printf("Press 4 to delete the entire list.\n");
      printf("Enter your choice: \n");
      scanf("%d", &choice);
      if (choice == 1) {
        delete_from_beginning(&head);
      } else if (choice == 2) {
        delete_from_end(&head);
      } else if (choice == 3) {
        printf("Enter the index that you want to delete from:\n");
        int index;
```

```
        scanf("%d", &index);
        delete_from_index(&head, index);
      } else if(choice == 4) {
        clear_list(&head);
      } else {
        printf("Wrong choice entered.\n");
      }
    } else if (choice == 3) {
      return 0;
    } else {
      printf("Wrong choice entered!");
    }
    print(head);
  }
  return 0;
}
```

Output:

```
Press 1 to add a node, and 2 to delete a node, and 3 to exit.
Enter your choice:
1
Press 1 to add node at beginning, 2 to add node at end, and 3 to add node after a
certain index.
Enter your choice:
1
Enter the value to be added to the list:
3
List is: 3
Press 1 to add a node, and 2 to delete a node, and 3 to exit.
Enter your choice:
1
Press 1 to add node at beginning, 2 to add node at end, and 3 to add node after a
certain index.
Enter your choice:
2
Enter the value to be added to the list:
5
List is: 3 5
Press 1 to add a node, and 2 to delete a node, and 3 to exit.
Enter your choice:
2
Press 1 to delete node from beginning, 2 to delete node from end, and 3 to delete node
from any index.
Press 4 to delete the entire list.
Enter your choice:
3
Enter the index that you want to delete from:
0
List is: 5
Press 1 to add a node, and 2 to delete a node, and 3 to exit.
Enter your choice:
3
```

Question 3. Write a program in C to reverse a singly linked list. You may create a header file of your own to perform the task.

Program:

```
#include "linked_list.h"
void reverse(node** head){
  node* nxt=NULL, *prev=NULL, *current=*head;
  while(current != NULL){
    nxt = current->next;
    current->next = prev;
    prev = current;
    current = nxt;
```

```
  }
  *head = prev;
}
int main(){
  node* head = NULL;
  while(1){
    printf("Press 1 to insert an element in the list.\n");
    printf("Press 2 to reverse the linked list.\n");
    printf("Press any other key to exit.\n");
    printf("Enter your choice:\n");
    int choice;
    scanf("%d",&choice);
    switch(choice){
      case 1:
        printf("Enter element to insert in list:\n");
        int ele;
        scanf("%d", &ele);
        insert_at_end(&head,ele);
        print(head);
        break;
      case 2:
        reverse(&head);
        print(head);
        break;
      default:
        exit(0);
    }
  }
  return 0;
}
```

Output:

```
Press 1 to insert an element in the list.
Press 2 to reverse the linked list.
Press any other key to exit.
Enter your choice:
1
Enter element to insert in list:
6
List is: 6
Press 1 to insert an element in the list.
Press 2 to reverse the linked list.
Press any other key to exit.
Enter your choice:
1
Enter element to insert in list:
8
List is: 6 8
Press 1 to insert an element in the list.
Press 2 to reverse the linked list.
Press any other key to exit.
Enter your choice:
1
Enter element to insert in list:
92
List is: 6 8 92
Press 1 to insert an element in the list.
Press 2 to reverse the linked list.
Press any other key to exit.
Enter your choice:
2
List is: 92 8 6
Press 1 to insert an element in the list.
Press 2 to reverse the linked list.
Press any other key to exit.
Enter your choice:
3
```

Question 4. Write a C program to design a Doubly Linked List and perform Insertion at end, Insertion at Beginning, Deletion at Begin, Deletion at End, Insertion at Index, Deletion at Index, Deletion of entire List, Reverse operation, and Display Linked List operation on it. You may create a header file of your own to perform the task.

Program:

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

typedef struct node{
  int data;
  struct node *previous;
  struct node *next;
} node;
node *create_node(int data){
  node *newnode = (node *)malloc(sizeof(node));
  newnode->data = data;
  newnode->previous = NULL;
  newnode->next = NULL;
  return newnode;
}
void print(node *head){
  printf("List is: ");
  for (node *tmp = head; tmp != NULL; tmp = tmp->next){
    printf("%d ", tmp->data);
  }
  printf("\n");
}
void insert_at_beginning(node** head,int data){
  node* newnode = create_node(data);
  if(*head == NULL){ *head = newnode; return; }
  newnode->next = *head;
  (*head)->previous = newnode;
  *head = newnode;
  return;
}
void insert_at_end(node** head,int data){
  if(*head == NULL) return;
  node* newnode = create_node(data);
  node* tmp = *head;
  int i = 0;
  for (; tmp->next != NULL; i++, tmp = tmp->next);
  newnode->next = tmp->next;
  newnode->previous = tmp;
  tmp->next = newnode;
}
void insert_at_index(node **head, int data, int pos){
  node *newnode = create_node(data);
  if (*head == NULL){
    *head = newnode;
    return;
  }
  if (pos == 0){ insert_at_beginning(head,data); return; }
  int i = 0;
  node *tmp = *head;
  for (; i < pos - 1 && tmp->next != NULL; i++, tmp = tmp->next);
  newnode->next = tmp->next;
  newnode->previous = tmp;
  tmp->next = newnode;
}

void delete_from_beginning(node** head){
  if (*head == NULL) return;
  node *temp = *head;
```

```
  if (temp->next != NULL) temp->previous = NULL;
  *head = (*head)->next;
  free(temp);
}
void delete_from_end(node** head){
  if (*head == NULL) return;
  int i = 0;
  node *tmp = *head;
  for (; tmp->next != NULL; i++, tmp = tmp->next);
  if (tmp->previous != NULL) tmp->previous->next = tmp->next;
  if (tmp->next != NULL) tmp->next->previous = tmp->previous;
  free(tmp);
}
void delete_from_index(node **head, int pos){
  if (*head == NULL) return;
  if (pos == 0){ delete_from_beginning(head); return; }
  int i = 0;
  node *tmp = *head;
  for (; i < pos && tmp->next != NULL; i++, tmp = tmp->next);
  if (tmp->previous != NULL) tmp->previous->next = tmp->next;
  if (tmp->next != NULL) tmp->next->previous = tmp->previous;
  free(tmp);
}
void clear_list(node** head){
  node* temp = *head;
  while(temp != NULL){
    *head=NULL;
    free(*head);
    (*head) = temp->next;
    temp = temp->next;
  }
}

void reverse(node **head){
  node *temp;
  for (node *tmp = *head; tmp != NULL; tmp = tmp->previous){
    if (tmp->next == NULL) temp = tmp;
    node *temp1 = tmp->next;
    tmp->next = tmp->previous;
    tmp->previous = temp1;
  }
  *head = temp;
}

#include "doubly_linked_list.h"
int main(){
  node *head = NULL;
  while(1){
    printf("Press 1 to insert node.\n");
    printf("Press 2 to delete node.\n");
    printf("Press 3 to reverse list.\n");
    printf("Press 4 to print list.\n");
    printf("Press any other number to exit.\n");
    printf("Enter your choice:\n");
    int choice;
    scanf("%d", &choice);
    switch (choice){
      int num, pos;
      case 1:
        printf("Enter the number you want to enter:\n");
        scanf("%d", &num);
        printf("Press 1 to enter at beginning, 2 to enter at end, 3 to enter at any index.\n");
        printf("Enter your choice:\n");
        int choice2;
        scanf("%d", &choice2);
        switch(choice2){
          case 1:
            insert_at_beginning(&head,num);
            break;
          case 2:
```

```
            insert_at_end(&head,num);
            break;
          case 3:
            printf("Enter the position (0-indexed) that you want to enter in:\n");
            scanf("%d", &pos);
            insert_at_index(&head, num, pos);
            break;
          default:
            exit(0);
        }
        break;
      case 2:
        printf("Press 1 to delete from beginning,2 to delete from end,3 to delete from any
index\n");
        printf("Press 4 to delete entire list.\n");
        printf("Enter your choice:\n");
        scanf("%d", &choice2);
        switch(choice2){
          case 1:
            delete_from_beginning(&head);
            break;
          case 2:
            delete_from_end(&head);
            break;
          case 3:
            printf("Enter the position (0-indexed) that you want to delete from:\n");
            scanf("%d", &pos);
            delete_from_index(&head, pos);
            break;
          case 4:
            clear_list(&head);
            break;
          default:
            exit(0);
        }
        break;
      case 3:
        reverse(&head);
        break;
      case 4:
        print(head);
        break;
      default:
        exit(0);
    }
  }
  return 0;
}
```

Output:

```
Press 1 to insert node.
Press 2 to delete node.
Press 3 to reverse list.
Press 4 to print list.
Press any other number to exit.
Enter your choice:
1
Enter the number you want to enter:
1
Press 1 to enter at beginning, 2 to enter at end, 3 to enter at any index.
Enter your choice:
1
Press 1 to insert node.
Press 2 to delete node.
Press 3 to reverse list.
Press 4 to print list.
Press any other number to exit.
Enter your choice:
1
```

```
Enter the number you want to enter:
6
Press 1 to enter at beginning, 2 to enter at end, 3 to enter at any index.
Enter your choice:
2
Press 1 to insert node.
Press 2 to delete node.
Press 3 to reverse list.
Press 4 to print list.
Press any other number to exit.
Enter your choice:
4
List is: 1 6
Press 1 to insert node.
Press 2 to delete node.
Press 3 to reverse list.
Press 4 to print list.
Press any other number to exit.
Enter your choice:
3
Press 1 to insert node.
Press 2 to delete node.
Press 3 to reverse list.
Press 4 to print list.
Press any other number to exit.
Enter your choice:
4
List is: 6 1
Press 1 to insert node.
Press 2 to delete node.
Press 3 to reverse list.
Press 4 to print list.
Press any other number to exit.
Enter your choice:
2
Press 1 to delete from beginning,2 to delete from end,3 to delete from any index
Press 4 to delete entire list.
Enter your choice:
2
Press 1 to insert node.
Press 2 to delete node.
Press 3 to reverse list.
Press 4 to print list.
Press any other number to exit.
Enter your choice:
4
List is: 6
Press 1 to insert node.
Press 2 to delete node.
Press 3 to reverse list.
Press 4 to print list.
Press any other number to exit.
Enter your choice:
5
```

Question 5. Write a C program to design a Circular Linked List and perform Insertion at end, Insertion at Beginning, Deletion at Begin, Deletion at End, Insertion at Index, Deletion at Index, Deletion of entire List, and Display Circular Linked List operation. You may create a header file of your own to perform the task.

Program:

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

struct node{
  int data;
  struct node* next;
};
```

```
typedef struct node node;

node* create_node(int data){
  node* newnode = (node*)malloc(sizeof(node));
  newnode->data = data;
  newnode->next = NULL;
  return newnode;
}
node* head;
void print(){
  if(head == NULL){
    printf("List is empty!\n");
    return;
  }
  printf("List is = %d ", head->data);
  for(node* tmp = head->next; tmp != head; tmp = tmp->next){
    printf("%d ", tmp->data);
  }
  printf("\n");
}

void insert_at_beginning(int data){
  node* newnode = create_node(data);
  if(head == NULL){
    newnode->next = newnode;
    head = newnode;
    return;
  }
  node* lastnode = head;
  while(lastnode->next != head) lastnode = lastnode->next;
  lastnode->next = newnode;
  newnode->next = head;
  head = newnode;
}
void insert_at_end(int data){
  node* newnode = create_node(data);
  if(head == NULL){
    newnode->next = newnode;
    head = newnode;
    return;
  }
  node* tmp = head;
  int i = 0;
  while(tmp->next != head){
    tmp = tmp->next; i++;
  }
  newnode->next = tmp->next;
  tmp->next = newnode;
}
void insert_at_index(int data,int pos){
  node* newnode = create_node(data);
  if(head == NULL){
    newnode->next = newnode;
    head = newnode;
    return;
  }
  if(pos == 0){ insert_at_beginning(data); return; }
  node* tmp = head;
  int i = 0;
  while(i < pos-1 && tmp->next != head){
    tmp = tmp->next; i++;
  }
  newnode->next = tmp->next;
  tmp->next = newnode;
}

void delete_from_beginning(){
  if(head == NULL) return;
  node* lastnode = head;
  while(lastnode->next != head) lastnode = lastnode->next;
```

```
  lastnode->next = head->next;
  node* tmp = head;
  head = head->next;
  free(tmp);
  return;
}
void delete_from_end(){
  if(head == NULL) return;
  node* tmp = head;
  int i = 0;
  while(tmp->next->next != head){
    tmp = tmp->next; i++;
  }
  node* todel = tmp->next;
  tmp->next = tmp->next->next;
  if(todel == head) head = head->next;
  free(todel);
}
void delete_from_index(int pos){
  if(head == NULL) return;
  if(pos == 0){ delete_from_beginning(); return; }
  if(head->next == head){
    head = NULL; return;
  }
  node* tmp = head;
  int i = 0;
  while(i < pos-1 && tmp->next->next != head){
    tmp = tmp->next; i++;
  }
  node* todel = tmp->next;
  tmp->next = tmp->next->next;
  if(todel == head) head = head->next;
  free(todel);
}
void clear_list(){
  if(head->next == head){
    head = NULL; return;
  }
  node* temp = head;
  while(temp != NULL){
    head = NULL;
    free(head);
    head = temp->next;
    temp = temp->next;
  }
}

#include "circular_linked_list.h"
int main(){
  while(1){
    printf("Press 1 to enter an element in the list.\n");
    printf("Press 2 to delete an element from the list.\n");
    printf("Press 3 to print the list.\n");
    printf("Press any other integer to exit.\n");
    printf("Enter your choice:\n");
    int choice;
    scanf("%d",&choice);
    switch (choice){
      int num, pos;
      case 1:
        printf("Enter the number you want to enter:\n");
        scanf("%d", &num);
        printf("Press 1 to enter at beginning, 2 to enter at end, 3 to enter at any index.\n");
        printf("Enter your choice:\n");
        int choice2;
        scanf("%d", &choice2);
        switch(choice2){
          case 1:
            insert_at_beginning(num);
            break;
```

```
          case 2:
            insert_at_end(num);
            break;
          case 3:
            printf("Enter the position (0-indexed) that you want to enter in:\n");
            scanf("%d", &pos);
            insert_at_index(num, pos);
            break;
          default:
            exit(0);
        }
        break;
      case 2:
        printf("Press 1 to delete from beginning,2 to delete from end,3 to delete from any
index\n");
        printf("Press 4 to delete entire list.\n");
        printf("Enter your choice:\n");
        scanf("%d", &choice2);
        switch(choice2){
          case 1:
            delete_from_beginning();
            break;
          case 2:
            delete_from_end();
            break;
          case 3:
            printf("Enter the position (0-indexed) that you want to delete from:\n");
            scanf("%d", &pos);
            delete_from_index(pos);
            break;
          case 4:
            clear_list();
            break;
          default:
            exit(0);
        }
        break;
      case 3:
        print();
        break;
      default:
        exit(0);
    }
  }
  return 0;
}
```

Output:

```
Press 1 to enter an element in the list.
Press 2 to delete an element from the list.
Press 3 to print the list.
Press any other integer to exit.
Enter your choice:
1
Enter the number you want to enter:
6
Press 1 to enter at beginning, 2 to enter at end, 3 to enter at any index.
Enter your choice:
1
Press 1 to enter an element in the list.
Press 2 to delete an element from the list.
Press 3 to print the list.
Press any other integer to exit.
Enter your choice:
1
Enter the number you want to enter:
8
Press 1 to enter at beginning, 2 to enter at end, 3 to enter at any index.
Enter your choice:
```

```
2
Press 1 to enter an element in the list.
Press 2 to delete an element from the list.
Press 3 to print the list.
Press any other integer to exit.
Enter your choice:
3
List is = 6 8
Press 1 to enter an element in the list.
Press 2 to delete an element from the list.
Press 3 to print the list.
Press any other integer to exit.
Enter your choice:
2
Press 1 to delete from beginning,2 to delete from end,3 to delete from any index
Press 4 to delete entire list.
Enter your choice:
1
Press 1 to enter an element in the list.
Press 2 to delete an element from the list.
Press 3 to print the list.
Press any other integer to exit.
Enter your choice:
3
List is = 8
Press 1 to enter an element in the list.
Press 2 to delete an element from the list.
Press 3 to print the list.
Press any other integer to exit.
Enter your choice:
5
```

Question 6. Write a C program to design a Circular Doubly Linked List and perform Insertion at end, Insertion at Beginning, Deletion at Begin, Deletion at End, Insertion at Index, Deletion at Index, Deletion of entire List, and Display Circular Doubly Linked List operation. You may create a header file of your own to perform the task.

Program:

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

typedef struct node{
  int data;
  struct node *previous;
  struct node *next;
} node;

node* create_node(int data){
  node *newnode = (node*)malloc(sizeof(node));
  newnode->data = data;
  newnode->previous = NULL;
  newnode->next = NULL;
}

void print(node *head){
  if (head == NULL){
    printf("List is empty!\n");
    return;
  }
  printf("Linked list is: ");
  node *tmp = head;
  for (; tmp->next != head; tmp = tmp->next){
    printf("%d ", tmp->data);
  }
  printf("%d\n", tmp->data);
}
```

```
void insert_at_beginning(int data,node** head){
  node *newnode = create_node(data);
  if (*head == NULL){
    newnode->previous = newnode;
    newnode->next = newnode;
    *head = newnode;
    return;
  }
  newnode->next = *head;
  newnode->previous = (*head)->previous;
  (*head)->previous = newnode;
  newnode->previous->next = newnode;
  *head = newnode;
}
void insert_at_end(int data,node** head){
  node *newnode = create_node(data);
  node *tmp = *head;
  int i = 0;
  for (tmp = *head; tmp->next != *head; tmp = tmp->next, i++);
  newnode->previous = tmp;
  newnode->next = tmp->next;
  tmp->next = newnode;
  newnode->next->previous = newnode;
}
void insert_at_index(int data, int position, node **head){
  node *newnode = create_node(data);
  if (*head == NULL){
    newnode->previous = newnode;
    newnode->next = newnode;
    *head = newnode;
    return;
  }
  if (position == 0){
    newnode->next = *head;
    newnode->previous = (*head)->previous;
    (*head)->previous = newnode;
    newnode->previous->next = newnode;
    *head = newnode;
    return;
  }
  node *tmp = *head;
  int i = 0;
  for (tmp = *head; tmp->next != *head && i < position - 1; tmp = tmp->next, i++);
  newnode->previous = tmp;
  newnode->next = tmp->next;
  tmp->next = newnode;
  newnode->next->previous = newnode;
}

void delete_from_beginning(node** head){
  if (*head == NULL) return;
  node *tmp = *head;
  int i = 0;
  tmp->next->previous = tmp->previous;
  tmp->previous->next = tmp->next;
  if (tmp == *head){
    if (tmp->next != tmp) *head = tmp->next;
    else *head = NULL;
  }
  free(tmp);
}
void delete_from_end(node** head){
  if (*head == NULL) return;
  node *tmp;
  int i = 0;
  for (tmp = *head; tmp->next != *head; tmp = tmp->next, i++);
  tmp->next->previous = tmp->previous;
  tmp->previous->next = tmp->next;
  if (tmp == *head){
    if (tmp->next != tmp) *head = tmp->next;
```

```
    else *head = NULL;
  }
  free(tmp);
}
void delete_from_index(int position, node **head){
  if (*head == NULL) return;
  node *tmp;
  int i = 0;
  for (tmp = *head; tmp->next != *head && i < position; tmp = tmp->next, i++);
  tmp->next->previous = tmp->previous;
  tmp->previous->next = tmp->next;
  if (tmp == *head){
    if (tmp->next != tmp) *head = tmp->next;
    else *head = NULL;
  }
  free(tmp);
}
void clear_list(node** head){
  if((*head)->next == *head){
    *head = NULL; return;
  }
  node* temp = *head;
  while(temp != NULL){
    *head = NULL;
    free(*head);
    *head = temp->next;
    temp = temp->next;
  }
}

#include "circular_doubly_linked_list.h"
int main(){
  node* head = NULL;
  while(1){
    printf("Press 1 to enter an element in the list.\n");
    printf("Press 2 to delete an element from the list.\n");
    printf("Press 3 to print the list.\n");
    printf("Press any other integer to exit.\n");
    printf("Enter your choice:\n");
    int choice;
    scanf("%d",&choice);
    switch (choice){
      int num, pos;
      case 1:
        printf("Enter the number you want to enter:\n");
        scanf("%d", &num);
        printf("Press 1 to enter at beginning, 2 to enter at end, 3 to enter at any index.\n");
        printf("Enter your choice:\n");
        int choice2;
        scanf("%d", &choice2);
        switch(choice2){
          case 1:
            insert_at_beginning(num,&head);
            break;
          case 2:
            insert_at_end(num,&head);
            break;
          case 3:
            printf("Enter the position (0-indexed) that you want to enter in:\n");
            scanf("%d", &pos);
            insert_at_index(num,pos,&head);
            break;
          default:
            exit(0);
        }
        break;
      case 2:
        printf("Press 1 to delete from beginning,2 to delete from end,3 to delete from any
index\n");
        printf("Press 4 to delete entire list.\n");
```

```
        printf("Enter your choice:\n");
        scanf("%d", &choice2);
        switch(choice2){
          case 1:
            delete_from_beginning(&head);
            break;
          case 2:
            delete_from_end(&head);
            break;
          case 3:
            printf("Enter the position (0-indexed) that you want to delete from:\n");
            scanf("%d", &pos);
            delete_from_index(pos,&head);
            break;
          case 4:
            clear_list(&head);
            break;
          default:
            exit(0);
        }
        break;
      case 3:
        print(head);
        break;
      default:
        exit(0);
    }
  }
  return 0;
}
```

Output:

```
Press 1 to enter an element in the list.
Press 2 to delete an element from the list.
Press 3 to print the list.
Press any other integer to exit.
Enter your choice:
1
Enter the number you want to enter:
6
Press 1 to enter at beginning, 2 to enter at end, 3 to enter at any index.
Enter your choice:
1
Press 1 to enter an element in the list.
Press 2 to delete an element from the list.
Press 3 to print the list.
Press any other integer to exit.
Enter your choice:
1
Enter the number you want to enter:
2
Press 1 to enter at beginning, 2 to enter at end, 3 to enter at any index.
Enter your choice:
2
Press 1 to enter an element in the list.
Press 2 to delete an element from the list.
Press 3 to print the list.
Press any other integer to exit.
Enter your choice:
3
Linked list is: 6 2
Press 1 to enter an element in the list.
Press 2 to delete an element from the list.
Press 3 to print the list.
Press any other integer to exit.
Enter your choice:
2
Press 1 to delete from beginning,2 to delete from end,3 to delete from any index
Press 4 to delete entire list.
```

```
Enter your choice:
3
Enter the position (0-indexed) that you want to delete from:
1
Press 1 to enter an element in the list.
Press 2 to delete an element from the list.
Press 3 to print the list.
Press any other integer to exit.
Enter your choice:
3
Linked list is: 6
Press 1 to enter an element in the list.
Press 2 to delete an element from the list.
Press 3 to print the list.
Press any other integer to exit.
Enter your choice:
5
```

Question 7. Write a C program to implement Last in First out (LIFO) data structure and perform Push, Pop, Peek operations on it. Use array to implement the same.

Program:

```
#include <stdio.h>
#include <stdlib.h>
#define MAX_SIZE 1005
int stack[MAX_SIZE];
int stackptr = -1;

void push(int data){
  if (stackptr == MAX_SIZE - 1){
    printf("Stack Overflow!\n");
    return;
  }
  stack[++stackptr] = data;
}
void pop(){
  if (stackptr == -1){
    printf("Stack Underflow!\n");
    return;
  }
  stackptr--;
}
void peek(){
  if (stackptr == -1){
    printf("Stack is empty!\n");
    return;
  }
  printf("Top element of stack = %d\n", stack[stackptr]);
}

int main(){
  while (1){
    printf("Press 1 to push element in stack.\n");
    printf("Press 2 to pop element in stack.\n");
    printf("Press 3 to view top element in stack.\n");
    printf("Press any other key to exit.\n");
    printf("Enter your choice:\n");
    int choice;
    scanf("%d", &choice);
    switch(choice){
    case 1:
      printf("Enter the element that you want to push in the stack:\n");
      int data;
      scanf("%d", &data);
      push(data);
```

```
      break;
    case 2:
      pop();
      break;
    case 3:
      peek();
      break;
    default:
      exit(0);
    }
  }
}
```

Output:

```
Press 1 to push element in stack.
Press 2 to pop element in stack.
Press 3 to view top element in stack.
Press any other key to exit.
Enter your choice:
1
Enter the element that you want to push in the stack:
2
Press 1 to push element in stack.
Press 2 to pop element in stack.
Press 3 to view top element in stack.
Press any other key to exit.
Enter your choice:
3
Top element of stack = 2
Press 1 to push element in stack.
Press 2 to pop element in stack.
Press 3 to view top element in stack.
Press any other key to exit.
Enter your choice:
1
Enter the element that you want to push in the stack:
8
Press 1 to push element in stack.
Press 2 to pop element in stack.
Press 3 to view top element in stack.
Press any other key to exit.
Enter your choice:
3
Top element of stack = 8
Press 1 to push element in stack.
Press 2 to pop element in stack.
Press 3 to view top element in stack.
Press any other key to exit.
Enter your choice:
2
Press 1 to push element in stack.
Press 2 to pop element in stack.
Press 3 to view top element in stack.
Press any other key to exit.
Enter your choice:
3
Top element of stack = 2
Press 1 to push element in stack.
Press 2 to pop element in stack.
Press 3 to view top element in stack.
Press any other key to exit.
Enter your choice:
2
Press 1 to push element in stack.
Press 2 to pop element in stack.
Press 3 to view top element in stack.
Press any other key to exit.
Enter your choice:
3
Stack is empty!
```

```
Press 1 to push element in stack.
Press 2 to pop element in stack.
Press 3 to view top element in stack.
Press any other key to exit.
Enter your choice:
5
```

Question 8. Write a C program to implement Push, Pop, Peek operations of Stack using Linked List.

Program:

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

typedef struct stack_node{
  int data;
  struct stack_node *previous;
} stack_node;

stack_node* create_node(int data){
  stack_node *newnode = (stack_node *)malloc(sizeof(stack_node));
  newnode->data = data;
  newnode->previous = NULL;
  return newnode;
}
void push(stack_node **head, int data){
  stack_node *newnode = create_node(data);
  if (*head == NULL){
    *head = newnode;
    return;
  }
  newnode->previous = *head;
  *head = newnode;
}
void pop(stack_node **head){
  if (*head == NULL) return;
  stack_node *tmp = *head;
  *head = (*head)->previous;
  free(tmp);
}
int peek(stack_node *head){
  if (head == NULL) return -1;
  return head->data;
}
int empty(stack_node *head){
  return head == NULL;
}
```

Output:

```
N/A
```

Question 9. Write a C program to reverse an array using stack. You need to implement the stack first then use that for reversal. You may create a header file of your own to perform the task.

Program:

```
#include "stack.h"
int main(){
  printf("Enter the number of elements in the array:\n");
  int n;
  scanf("%d",&n);
  printf("Enter %d elements:\n", n);
  int* a = (int*)malloc(sizeof(int)*n);
```

```
  for(int i=0; i<n; i++) scanf("%d", &a[i]);
  stack_node* head = NULL;
  for(int i=0; i<n; i++) push(&head,a[i]);
  for(int i=0; i<n; i++){ a[i] = peek(head); pop(&head); }
  printf("Reversed array = ");
  for(int i=0; i<n; i++) printf("%d ",a[i]);
  printf("\n");
  return 0;
}
```

Output:

```
Enter the number of elements in the array:
5
Enter 5 elements:
3 5 9 7 8
Reversed array = 8 7 9 5 3
```

Question 10. Write a C program to convert an equation to Postfix and Prefix with the help of a Stack. Implement the Stack using Linked List.

Program:

```
#include "stack.h"
#include <string.h>

int precedence(char ch){
  if (ch == '+' || ch == '-') return 0;
  else if (ch == '*' || ch == '/') return 1;
  else if (ch == '^') return 2;
  else if (ch == '(' || ch == ')') return 3;
  else return -1;
}
char* reverse(char *str){
  int len = strlen(str);
  char *res = (char *)malloc(sizeof(char) * len);
  for (int i = 0; i < len; i++) res[i] = str[len - 1 - i];
  return res;
}
char * convert_to_postfix(char *expr){
  char *res = (char *)malloc(sizeof(char) * strlen(expr) + 5);
  for (int i = 0; i < strlen(res); i++) res[i] = NULL;
  int res_idx = 0;
  stack_node *stk = NULL;
  for (int i = 0; i < strlen(expr); i++){
    char ch = expr[i];
    int pr = precedence(ch);
    if (pr == -1)res[res_idx++] = ch;
    else{
      if (ch == ')'){
        while (!empty(stk)){
          char ch = peek(stk);
          pop(&stk);
          if (ch == '(') break;
          else res[res_idx++] = ch;
        }
      }else{
        while (!empty(stk)){
          if (peek(stk) == '(') break;
          if (precedence(peek(stk)) > pr){
            res[res_idx++] = peek(stk);
            pop(&stk);
          }else if (precedence(peek(stk)) == pr){
            if (ch == '^' || ch == '(') break;
            res[res_idx++] = peek(stk);
            pop(&stk);
```

```
          } else break;
        }
        push(&stk, ch);
      }
    }
  }
  while (!empty(stk)){
    if (peek(stk) != '(') res[res_idx++] = peek(stk);
    pop(&stk);
  }
  return res;
}
char * convert_to_prefix(char *expr){
  char *res = reverse(expr);
  for (int i = 0; i < strlen(res); i++){
    if (res[i] == '(') res[i] = ')';
    else if (res[i] == ')') res[i] = '(';
  }
  res = convert_to_postfix(res);
  res = reverse(res);
  return res;
}
int main(){
  while (1){
    printf("Enter expression (without spaces):\n");
    char str[100] = {NULL};
    scanf("%s", str);
    printf("Press 1 to convert to postfix, 2 to convert to prefix, any other integer to
exit.\n");
    printf("Enter your choice:");
    int choice;
    scanf("%d", &choice);
    switch (choice){
    case 1:
      printf("Postfix expression = %s\n", convert_to_postfix(str));
      break;
    case 2:
      printf("Prefix expression = %s\n", convert_to_prefix(str));
      break;
    default:
      exit(0);
    }
  }
  return 0;
}
```

Output:

```
Enter expression (without spaces):
(a+b)*(c-(d^e)+f)
Press 1 to convert to postfix, 2 to convert to prefix, any other integer to exit.
Enter your choice:1
Postfix expression = ab+cde^-f+*
```

Question 11. Write a C program to perform First in First out Data Structure. You need to perform Enqueue and Dequeue operation on this data structure. You may create a header file of your own to perform the task.

Program:

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 100001
int queue[MAX];
int front = -1, rear = -1;

void push(int data){
```

```
  if (rear == MAX - 1){
    printf("Queue is full!\n");
    return;
  }
  queue[++rear] = data;
  front = (front == -1 ? rear : front);
}
void pop(){
  if (front == -1){
    printf("Queue in empty!\n");
    return;
  }
  front++; front = (front > rear ? -1 : front);
}
int top(){
  if (front == -1){
    printf("Queue in empty!\n");
    return -1;
  }
  return queue[front];
}
void print(){
  if (front == -1){
    printf("Queue in empty!\n");
    return;
  }
  printf("Queue is : ");
  for (int i = front; i <= rear; i++) printf("%d ", queue[i]);
  printf("\n");
}
int empty(){ return front == -1; }

#include "queue.h"
int main(){
  while (1){
    printf("Press 1 to insert element, 2 to pop element from queue.\n");
    printf("Press 3 to print the queue, 4 to print the topmost element.\n");
    printf("Press any other integer to exit.\n");
    printf("Enter your choice:\n");
    int choice;
    scanf("%d", &choice);
    switch (choice){
    case 1:
      printf("Enter data to be added to queue:\n");
      int data;
      scanf("%d", &data);
      push(data);
      break;
    case 2:
      pop();
      break;
    case 3:
      print();
      break;
    case 4:
      printf("Topmost element = %d\n", top());
      break;
    default:
      exit(0);
    }
  }
  return 0;
}
```

Output:

```
Press 1 to insert element, 2 to pop element from queue.
Press 3 to print the queue, 4 to print the topmost element.
Press any other integer to exit.
```

```
Enter your choice:
1
Enter data to be added to queue:
3
Press 1 to insert element, 2 to pop element from queue.
Press 3 to print the queue, 4 to print the topmost element.
Press any other integer to exit.
Enter your choice:
1
Enter data to be added to queue:
6
Press 1 to insert element, 2 to pop element from queue.
Press 3 to print the queue, 4 to print the topmost element.
Press any other integer to exit.
Enter your choice:
3
Queue is : 3 6
Press 1 to insert element, 2 to pop element from queue.
Press 3 to print the queue, 4 to print the topmost element.
Press any other integer to exit.
Enter your choice:
2
Press 1 to insert element, 2 to pop element from queue.
Press 3 to print the queue, 4 to print the topmost element.
Press any other integer to exit.
Enter your choice:
4
Topmost element = 6
Press 1 to insert element, 2 to pop element from queue.
Press 3 to print the queue, 4 to print the topmost element.
Press any other integer to exit.
Enter your choice:
2
Press 1 to insert element, 2 to pop element from queue.
Press 3 to print the queue, 4 to print the topmost element.
Press any other integer to exit.
Enter your choice:
3
Queue in empty!
Press 1 to insert element, 2 to pop element from queue.
Press 3 to print the queue, 4 to print the topmost element.
Press any other integer to exit.
Enter your choice:
5
```

Question 12. Write a C program to implement circular queue using array. You need to perform Enqueue and Dequeue operation on this data structure. You may create a header file of your own to perform the task.

Program:

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 1001
int queue[MAX];
int tail=-1, head=-1;

void push(int data){
  if(head == -1){
    head = tail = 0;
    queue[head] = data;
    return;
  }
  if((tail+1)%MAX != head){
    tail = (tail+1)%MAX;
    queue[tail] = data;
  }else{
    printf("Queue is full!\n");
  }
```

```
}
void pop(){
  if(head == -1) return;
  head = (head+1)%MAX;
  if(head > tail) head = tail = -1;
}
int peek(){
  if(head == -1){
    printf("Queue is empty!\n");
    return -1;
  }
  return queue[head];
}
#include "circular_queue.h"
int main(){
  while(1){
    printf("Press 1 to insert an element, 2 to pop an element, 3 to peek.\n");
    printf("Press any other integer to exit.\n");
    printf("Enter your choice:\n");
    int choice;
    scanf("%d", &choice);
    switch(choice){
      case 1:
        printf("Enter element to insert in queue:\n");
        int ele;
        scanf("%d", &ele);
        push(ele);
        break;
      case 2:
        pop();
        break;
      case 3:
        printf("Top element = %d\n", peek());
        break;
      default:
        exit(0);
    }
  }
  return 0;
}
```

Output:

```
Press 1 to insert an element, 2 to pop an element, 3 to peek.
Press any other integer to exit.
Enter your choice:
1
Enter element to insert in queue:
12
Press 1 to insert an element, 2 to pop an element, 3 to peek.
Press any other integer to exit.
Enter your choice:
3
Top element = 12
Press 1 to insert an element, 2 to pop an element, 3 to peek.
Press any other integer to exit.
Enter your choice:
1
Enter element to insert in queue:
66
Press 1 to insert an element, 2 to pop an element, 3 to peek.
Press any other integer to exit.
Enter your choice:
3
Top element = 12
Press 1 to insert an element, 2 to pop an element, 3 to peek.
Press any other integer to exit.
Enter your choice:
2
Press 1 to insert an element, 2 to pop an element, 3 to peek.
```

```
Press any other integer to exit.
Enter your choice:
3
Top element = 66
Press 1 to insert an element, 2 to pop an element, 3 to peek.
Press any other integer to exit.
Enter your choice:
5
```

Question 13. Write a C program to implement circular queue using linked list. You need to perform Enqueue and Dequeue operation on this data structure. You may create a header file of your own to perform the task.

Program:

```
#include <stdio.h>
#include <stdlib.h>
typedef struct node{
  int data;
  struct node* next;
} node;
node* head=NULL, *tail=NULL;
node* create_node(int data){
  node* newnode = (node*)malloc(sizeof(node));
  newnode->data = data;
  newnode->next = NULL;
}

void push(int data){
  node* newnode = (node*)malloc(sizeof(node));
  newnode->data = data;
  if(head == NULL){
    head = tail = newnode;
    head->next = tail;
    tail->next = head;
    return;
  }
  tail->next = newnode;
  newnode->next = head;
  tail = newnode;
}
void pop(){
  if(head == NULL) return;
  node* tmp = head;
  if(head == tail) head = NULL;
  else head = head->next;
  free(tmp);
  if(head == NULL) tail=NULL;
}
int peek(){
  if(head == NULL){
    printf("Queue is empty!\n");
    return -1;
  }
  return head->data;
}
#include "circular_queue.h"
int main(){
  while(1){
    printf("Press 1 to insert an element, 2 to pop an element, 3 to peek.\n");
    printf("Press any other integer to exit.\n");
    printf("Enter your choice:\n");
    int choice;
    scanf("%d", &choice);
    switch(choice){
      case 1:
```

```
        printf("Enter element to insert in queue:\n");
        int ele;
        scanf("%d", &ele);
        push(ele);
        break;
      case 2:
        pop();
        break;
      case 3:
        printf("Top element = %d\n", peek());
        break;
      default:
        exit(0);
    }
  }
  return 0;
}
```

Output:

```
Press 1 to insert an element, 2 to pop an element, 3 to peek.
Press any other integer to exit.
Enter your choice:
1
Enter element to insert in queue:
6
Press 1 to insert an element, 2 to pop an element, 3 to peek.
Press any other integer to exit.
Enter your choice:
3
Top element = 6
Press 1 to insert an element, 2 to pop an element, 3 to peek.
Press any other integer to exit.
Enter your choice:
1
Enter element to insert in queue:
8
Press 1 to insert an element, 2 to pop an element, 3 to peek.
Press any other integer to exit.
Enter your choice:
3
Top element = 6
Press 1 to insert an element, 2 to pop an element, 3 to peek.
Press any other integer to exit.
Enter your choice:
2
Press 1 to insert an element, 2 to pop an element, 3 to peek.
Press any other integer to exit.
Enter your choice:
3
Top element = 8
Press 1 to insert an element, 2 to pop an element, 3 to peek.
Press any other integer to exit.
Enter your choice:
5
```

Question 14. Write a C program to implement Deque using array. Perform the operations add to front, add to rear, delete from front, delete from rear, and display. You may create a header file of your own to perform the task.

Program:

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 101
int deque[MAX];
int head=-1, tail=-1;
```

```
void push_back(int data){
  if(head == -1){
    head = tail = 0;
    deque[head] = data;
    return;
  }
  if((tail+1)%MAX != head){
    tail = (tail+1)%MAX;
    deque[tail] = data;
  }else{
    printf("Queue is full!\n");
  }
}
void push_front(int data){
  if(head == -1){
    head = tail = 0;
    deque[head] = data;
    return;
  }
  if((head-1+MAX)%MAX != tail){
    head = (head-1+MAX)%MAX;
    deque[head] = data;
  }else{
    printf("Queue is full!\n");
  }
}
void pop_front(){
  if(head == -1) return;
  head = (head+1)%MAX;
  if(head > tail) head = tail = -1;
}
void pop_back(){
  if(head == -1) return;
  tail = (tail-1+MAX)%MAX;
  if(head > tail) head = tail = -1;
}
int peek_front(){
  if(head == -1){
    printf("Deque is empty!\n");
    return -1;
  }else return deque[head];
}
int peek_back(){
  if(head == -1){
    printf("Deque is empty!\n");
    return -1;
  }else return deque[tail];
}
#include "sdeque.h"
int main(){
  while(1){
    printf("Press 1 to insert at back, 2 to insert at front.\n");
    printf("Press 3 to delete from back, 4 to delete from front.\n");
    printf("Press 5 to peek last element, 6 to peek first element.\n");
    printf("Press any other integer to exit.\n");
    printf("Enter your choice:\n");
    int choice;
    scanf("%d",&choice);
    if(choice == 1 || choice == 2){
      printf("Enter element to insert in queue:\n");
      int ele;
      scanf("%d", &ele);
      if(choice == 1) push_back(ele);
      else push_front(ele);
    }else if(choice == 3 || choice == 4){
      if(choice == 3) pop_back();
      else pop_front();
    }else if(choice == 5 || choice == 6){
      if(choice == 5) printf("Top element = %d\n", peek_back());
      else printf("Top element = %d\n", peek_front());
```

```
    }else{
      exit(0);
    }
  }
  return 0;
}
```

Output:

```
Press 1 to insert at back, 2 to insert at front.
Press 3 to delete from back, 4 to delete from front.
Press 5 to peek last element, 6 to peek first element.
Press any other integer to exit.
Enter your choice:
1
Enter element to insert in queue:
5
Press 1 to insert at back, 2 to insert at front.
Press 3 to delete from back, 4 to delete from front.
Press 5 to peek last element, 6 to peek first element.
Press any other integer to exit.
Enter your choice:
2
Enter element to insert in queue:
7
Press 1 to insert at back, 2 to insert at front.
Press 3 to delete from back, 4 to delete from front.
Press 5 to peek last element, 6 to peek first element.
Press any other integer to exit.
Enter your choice:
5
Top element = 5
Press 1 to insert at back, 2 to insert at front.
Press 3 to delete from back, 4 to delete from front.
Press 5 to peek last element, 6 to peek first element.
Press any other integer to exit.
Enter your choice:
6
Top element = 7
Press 1 to insert at back, 2 to insert at front.
Press 3 to delete from back, 4 to delete from front.
Press 5 to peek last element, 6 to peek first element.
Press any other integer to exit.
Enter your choice:
3
Press 1 to insert at back, 2 to insert at front.
Press 3 to delete from back, 4 to delete from front.
Press 5 to peek last element, 6 to peek first element.
Press any other integer to exit.
Enter your choice:
6
Top element = 7
Press 1 to insert at back, 2 to insert at front.
Press 3 to delete from back, 4 to delete from front.
Press 5 to peek last element, 6 to peek first element.
Press any other integer to exit.
Enter your choice:
8
```

Question 15. Write a C program to implement Deque using linked list. Perform the operations add to front, add to rear, delete from front, delete from rear, and display. You may create a header file of your own to perform the task.

Program:

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

```
typedef struct node{
  int data;
  struct node* next;
  struct node* prev;
}node;
node* head=NULL, *tail=NULL;
node* create_node(int data){
  node* newnode = (node*)malloc(sizeof(node));
  newnode->data = data;
  newnode->next = NULL;
  newnode->prev = NULL;
}

void push_back(int data){
  node* newnode = create_node(data);
  if(head == NULL){
    head = tail = newnode;
    return;
  }
  tail->next = newnode;
  tail = newnode;
}
void push_front(int data){
  node* newnode = create_node(data);
  if(head == NULL){
    head = tail = newnode;
    return;
  }
  head->prev = newnode;
  head = newnode;
}

void pop_back(){
  if(head == NULL) return;
  node* tmp = tail;
  tail = tail->prev;
  free(tmp);
  if(tail == NULL) head = NULL;
}
void pop_front(){
  if(head == NULL) return;
  node* tmp = tail;
  tail = tail->prev;
  free(tmp);
  if(tail == NULL) head = NULL;
}

int peek_back(){
  if(head == NULL){
    printf("Deque is empty!\n");
    return -1;
  }
  return tail->data;
}
int peek_front(){
  if(head == NULL){
    printf("Deque is empty!\n");
    return -1;
  }
  return head->data;
}
#include "deque.h"
int main(){
  while(1){
    printf("Press 1 to insert at back, 2 to insert at front.\n");
    printf("Press 3 to delete from back, 4 to delete from front.\n");
    printf("Press 5 to peek last element, 6 to peek first element.\n");
    printf("Press any other integer to exit.\n");
    printf("Enter your choice:\n");
```

```
    int choice;
    scanf("%d",&choice);
    if(choice == 1 || choice == 2){
      printf("Enter element to insert in queue:\n");
      int ele;
      scanf("%d", &ele);
      if(choice == 1) push_back(ele);
      else push_front(ele);
    }else if(choice == 3 || choice == 4){
      if(choice == 3) pop_back();
      else pop_front();
    }else if(choice == 5 || choice == 6){
      if(choice == 5) printf("Top element = %d\n", peek_back());
      else printf("Top element = %d\n", peek_front());
    }else{
      exit(0);
    }
  }
}
```

Output:

```
Press 1 to insert at back, 2 to insert at front.
Press 3 to delete from back, 4 to delete from front.
Press 5 to peek last element, 6 to peek first element.
Press any other integer to exit.
Enter your choice:
1
Enter element to insert in queue:
6
Press 1 to insert at back, 2 to insert at front.
Press 3 to delete from back, 4 to delete from front.
Press 5 to peek last element, 6 to peek first element.
Press any other integer to exit.
Enter your choice:
2
Enter element to insert in queue:
20
Press 1 to insert at back, 2 to insert at front.
Press 3 to delete from back, 4 to delete from front.
Press 5 to peek last element, 6 to peek first element.
Press any other integer to exit.
Enter your choice:
5
Top element = 6
Press 1 to insert at back, 2 to insert at front.
Press 3 to delete from back, 4 to delete from front.
Press 5 to peek last element, 6 to peek first element.
Press any other integer to exit.
Enter your choice:
6
Top element = 20
Press 1 to insert at back, 2 to insert at front.
Press 3 to delete from back, 4 to delete from front.
Press 5 to peek last element, 6 to peek first element.
Press any other integer to exit.
Enter your choice:
4
Press 1 to insert at back, 2 to insert at front.
Press 3 to delete from back, 4 to delete from front.
Press 5 to peek last element, 6 to peek first element.
Press any other integer to exit.
Enter your choice:
5
Top element = 20
Press 1 to insert at back, 2 to insert at front.
Press 3 to delete from back, 4 to delete from front.
Press 5 to peek last element, 6 to peek first element.
Press any other integer to exit.
Enter your choice:
8
```

Question 16. Write a C program to implement a Binary Tree and perform Add Node, Delete Node, and Breadth First Traversal operation on it. You may create a header file of your own to perform the task.

Program:

```
#include <stdio.h>
#include <stdlib.h>
#define MAX_NODES 100

typedef struct node{
  int data;
  struct node* lchild;
  struct node* rchild;
} node;
int n;
int adj_mat[MAX_NODES][MAX_NODES];
int children[MAX_NODES];
node* create_node(int num){
  node* newnode = (node*)malloc(sizeof(node));
  newnode->data = num;
  newnode->lchild = NULL;
  newnode->rchild = NULL;
}
node* create_tree(int node_idx){
  node* newnode = create_node(node_idx);
  for(int i = 1; i <= n; i++){
    if(adj_mat[node_idx][i]){
      node* child = create_tree(i);
      if(newnode->lchild == NULL) newnode->lchild = child;
      else newnode->rchild = child;
    }
  }
  return newnode;
}
void print_tree(node* curnode){
  printf("curnode = %d\n", curnode->data);
  if(curnode->lchild != NULL){
    printf("left child of %d is %d\n", curnode->data,curnode->lchild->data);
  }
  if(curnode->rchild != NULL){
    printf("right child of %d is %d\n", curnode->data,curnode->rchild->data);
  }
  if(curnode->lchild != NULL) print_tree(curnode->lchild);
  if(curnode->rchild != NULL) print_tree(curnode->rchild);
}
int main(){
  printf("Enter the number of nodes in the tree:\n");
  scanf("%d",&n);
  printf("Enter the edges in the binary tree:\n");
  for(int i = 0; i < n-1; i++){
    while(1){
      printf("Enter the nodes to be connected with an edge:\n");
      int u,v;
      scanf("%d%d",&u,&v);
      if(u < 1 || v < 1 || u > n || v > n){
        printf("Invalid node number. Please enter again.\n");
    }else if(children[u] == 2){
        printf("%d already has 2 children. Please enter again.\n", u);
      }else{
        children[u]++;
        adj_mat[u][v] = 1;
        break;
      }
    }
  }
  node* root = create_tree(1);
  print_tree(root);
}
```

Output:

```
Enter the number of nodes in the tree:
5
Enter the edges in the binary tree:
Enter the nodes to be connected with an edge:
1 2
Enter the nodes to be connected with an edge:
1 5
Enter the nodes to be connected with an edge:
2 3
Enter the nodes to be connected with an edge:
3 4
curnode = 1
left child of 1 is 2
right child of 1 is 5
curnode = 2
left child of 2 is 3
curnode = 3
left child of 3 is 4
curnode = 4
curnode = 5
```

Question 17. Write a C program to implement a Binary Search Tree and perform Add Node, Delete Node. Also perform the Inorder, Preorder, and Postorder traversal on it. You may create a header file of your own to perform the task.

Program:

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

typedef struct bst_node{
  int data;
  struct bst_node* lchild;
  struct bst_node* rchild;
} node;
node* root = NULL;
node* createnode(int data){
  node* newnode = (node*)malloc(sizeof(node));
  newnode->lchild = NULL;
  newnode->rchild = NULL;
  newnode->data = data;
  return newnode;
}
void insert_node(int data){
  if(root == NULL){
    root = createnode(data);
    return;
  }
  node* curnode = root;
  while(1){
    if(data < curnode->data){
      if(curnode->lchild == NULL) break;
      else curnode = curnode->lchild;
    }else if(data > curnode->data){
      if(curnode->rchild == NULL) break;
      else curnode = curnode->rchild;
    }else return;
  }
  node* newnode = createnode(data);
  if(newnode->data < curnode->data) curnode->lchild = newnode;
  else curnode->rchild = newnode;
}
node* find_largest(node* curnode){
  if(curnode->rchild->rchild == NULL) return curnode;
```

```
  else return find_largest(curnode->rchild);
}
node* find_smallest(node* curnode){
  if(curnode->lchild->lchild == NULL) return curnode;
  else return find_smallest(curnode->lchild);
}
void delete_node(int data){
  if(root == NULL) return;
  node* curnode = root;
  while(curnode != NULL){
    if(curnode->data == data) break;
    else if(curnode->data < data) curnode = curnode->rchild;
    else curnode = curnode->lchild;
  }
  if(curnode == NULL) return;
  if(curnode->lchild == NULL && curnode->rchild == NULL){
    if(curnode == root){
      root = NULL;
      free(curnode);
    }
    else{
      node* curnode2 = root;
      while(curnode2 != NULL){
        if(curnode2->data < data){
          if(curnode2->rchild == curnode) break;
          else curnode2 = curnode2->rchild;
        }else{
          if(curnode2->lchild == curnode) break;
          else curnode2 = curnode2->lchild;
        }
      }
      if(curnode2->rchild == curnode){
        curnode2->rchild = NULL;
        free(curnode);
      }else{
        curnode2->lchild = NULL;
        free(curnode);
      }
    }
    return;
  }
  if(curnode->lchild != NULL){
    // find largest node in left subtree's parent
    if(curnode->lchild->rchild == NULL){
      curnode->data = curnode->lchild->data;
      curnode->lchild = NULL;
      return;
    }
    node* todel = find_largest(curnode->lchild);
    curnode->data = todel->rchild->data;
    todel->rchild = NULL;
  }
  else{
    // find smallest node in right subtree's parent
    if(curnode->rchild->lchild == NULL){
      curnode->data = curnode->rchild->data;
      curnode->rchild = NULL;
      return;
    }
    node* todel = find_smallest(curnode->rchild);
    curnode->data = todel->lchild->data;
    todel->lchild = NULL;
  }
}
void print_tree(node* curnode){
  // inorder traversal
  if(curnode == NULL) return;
  printf("%d ", curnode->data);
  print_tree(curnode->lchild);
  print_tree(curnode->rchild);
```

```
}
int main(){
  printf("Press 1 to insert data and 2 to delete data from tree.\n");
  printf("Press any other key to exit.\n");
  while(1){
    printf("Enter your choice (1 or 2):\n");
    int choice;
    scanf("%d", &choice);
    switch(choice){
      case 1:
        printf("Enter data to insert:\n");
        int data;
        scanf("%d", &data);
        insert_node(data);
        printf("Current state of tree = ");
        print_tree(root);
        printf("\n");
        break;
      case 2:
        printf("Enter data to remove:\n");
        int data;
        scanf("%d", &data);
        delete_node(data);
        printf("Current state of tree = ");
        print_tree(root);
        printf("\n");
        break;
      default:
        exit(0);
    }
  }
  return 0;
}
```

Output:

```
Press 1 to insert data and 2 to delete data from tree.
Press any other key to exit.
Enter your choice (1 or 2):
1
Enter data to insert:
6
Current state of tree (inorder) = 6
Enter your choice (1 or 2):
1
Enter data to insert:
8
Current state of tree (inorder) = 6 8
Enter your choice (1 or 2):
1
Enter data to insert:
15
Current state of tree (inorder) = 6 8 15
Enter your choice (1 or 2):
1
Enter data to insert:
3
Current state of tree (inorder) = 6 3 8 15
Enter your choice (1 or 2):
1
Enter data to insert:
12
Current state of tree (inorder) = 6 3 8 15 12
Enter your choice (1 or 2):
2
Enter data to remove:
3
Current state of tree (inorder) = 6 8 15 12
Enter your choice (1 or 2):
2
```

```
Enter data to remove:
15
Current state of tree (inorder) = 6 8 12
Enter your choice (1 or 2):
1
Enter data to insert:
1
Current state of tree (inorder) = 6 1 8 12
Enter your choice (1 or 2):
3
```

Question 18. Write a C program to sort an array using bubble sort technique.

Program:

```
#include <stdio.h>
#include <stdlib.h>
int main(){
  printf("Enter number of elements in the array:\n");
  int n;
  scanf("%d", &n);
  int* a = (int*)malloc(sizeof(int)*n);
  printf("Enter the elements of the array:\n");
  for(int i = 0; i < n; i++) scanf("%d", &a[i]);
  for(int i=0; i<n; i++){
    for(int j=0; j<n-i-1; j++){
      if(a[j] > a[j+1]){
        int temp = a[j]; a[j] = a[j+1]; a[j+1] = temp;
      }
    }
  }
  printf("Sorted array = ");
  for(int i = 0; i < n; i++) printf("%d ", a[i]);
  printf("\n");
  return 0;
}
```

Output:

```
Enter number of elements in the array:
5
Enter the elements of the array:
6 3 9 2 8
Sorted array = 2 3 6 8 9
```

Question 19. Write a C program to sort an array using selection sort technique.

Program:

```
#include <stdio.h>
#include <stdlib.h>
int main(){
  printf("Enter number of elements in the array:\n");
  int n;
  scanf("%d", &n);
  int* a = (int*)malloc(sizeof(int)*n);
  printf("Enter the elements of the array:\n");
  for(int i = 0; i < n; i++) scanf("%d", &a[i]);
  for(int i = 0; i < n; i++){
    int minidx = i;
    for(int j = i+1; j < n; j++){
      if(a[j] < a[minidx]) minidx = j;
    }
    int temp = a[minidx];
```

```
    a[minidx] = a[i];
    a[i] = temp;
  }
  printf("Sorted array = ");
  for(int i = 0; i < n; i++) printf("%d ", a[i]);
  printf("\n");
  return 0;
}
```

Output:

```
Enter number of elements in the array:
5
Enter the elements of the array:
12 5 1 85 2
Sorted array = 1 2 5 12 85
```

Question 20. Write a C program to sort an array using insertion sort technique.

Program:

```
#include <stdio.h>
#include <stdlib.h>
int main(){
  printf("Enter number of elements in the array:\n");
  int n;
  scanf("%d", &n);
  int* a = (int*)malloc(sizeof(int)*n);
  printf("Enter the elements of the array:\n");
  for(int i = 0; i < n; i++) scanf("%d", &a[i]);
  for(int i = 1; i < n; i++){
    int key = a[i];
    int j = i - 1;
    while (j >= 0 && a[j] > key){
      a[j + 1] = a[j];
      j = j - 1;
    }
    a[j + 1] = key;
  }
  printf("Sorted array = ");
  for(int i = 0; i < n; i++) printf("%d ", a[i]);
  printf("\n");
  return 0;
}
```

Output:

```
Enter number of elements in the array:
5
Enter the elements of the array:
7 45 6 12 5
Sorted array = 5 6 7 12 45
```

Question 21. Write a C program to sort an array using shell sort technique.

Program:

```
#include <stdio.h>
#include <stdlib.h>
int main(){
  printf("Enter number of elements in the array:\n");
  int n;
  scanf("%d", &n);
```

```
  int* a = (int*)malloc(sizeof(int)*n);
  printf("Enter the elements of the array:\n");
  for(int i = 0; i < n; i++) scanf("%d", &a[i]);
  for(int gap = n/2; gap > 0; gap /= 2){
    for(int i = gap; i < n; i += 1){
      int temp = a[i],j;
      for(j=i;j>=gap&&a[j-gap]>temp; j-=gap) a[j] = a[j-gap];
      a[j] = temp;
    }
  }
  printf("Sorted array = ");
  for(int i = 0; i < n; i++) printf("%d ", a[i]);
  printf("\n");
  return 0;
}
```

Output:

```
Enter number of elements in the array:
5
Enter the elements of the array:
6 9 3 68 2
Sorted array = 2 3 6 9 68
```