PRACTICAL – 04

Q. Write a program for creation of Linked list.

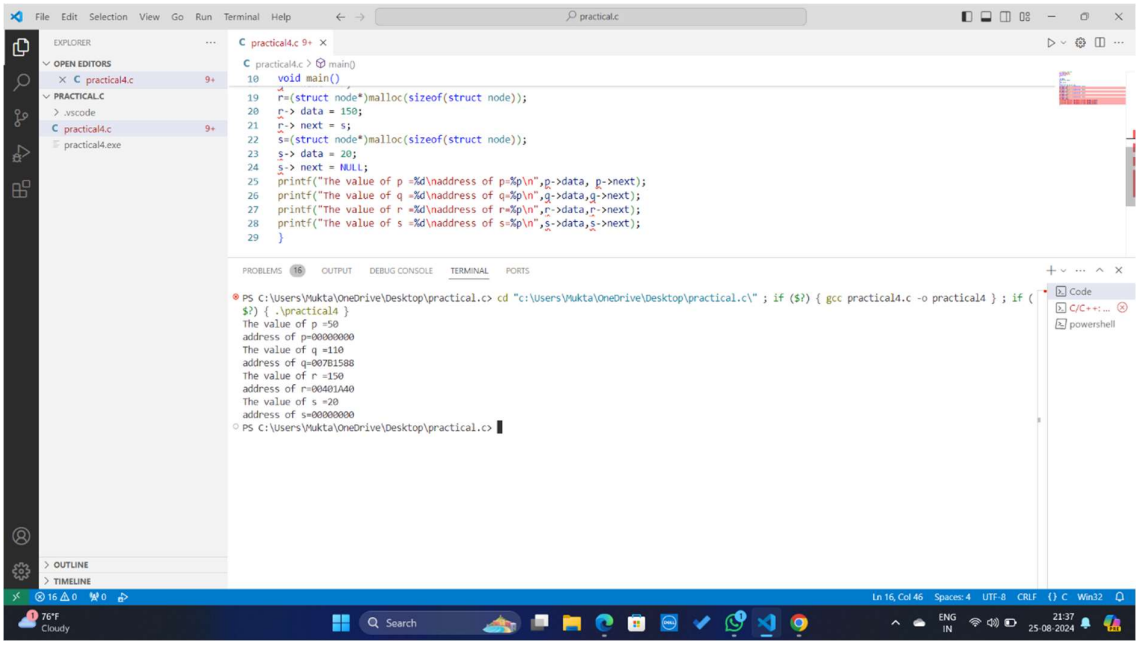
Program:

```
// create a link list
#include<stdio.h>
#include<malloc.h>
struct node
{
};
int data;
struct node * next;
void main()
{
struct node *p, *q ,*r ,*s;
p=(struct node*)malloc(sizeof(struct node));
p-> data = 5;
p-> next = q;
q=(struct node*)malloc(sizeof(struct node));
q-> data = 10;
q-> next = r;
r=(struct node*)malloc(sizeof(struct node));
r-> data = 15;
r-> next = s;
s=(struct node*)malloc(sizeof(struct node));
s-> data = 20;
s-> next = NULL;
printf("The value of p =%d\naddress of p=%p\n",p->data, p->next);
printf("The value of q =%d\naddress of q=%p\n",q->data,q->next);
printf("The value of r =%d\naddress of r=%p\n",r->data,r->next);
```

printf("The value of s =%d\naddress of s=%p\n",s->data,s->next);

}

Output:



Q. Write a program to insert an element at the beginning of Linked list.

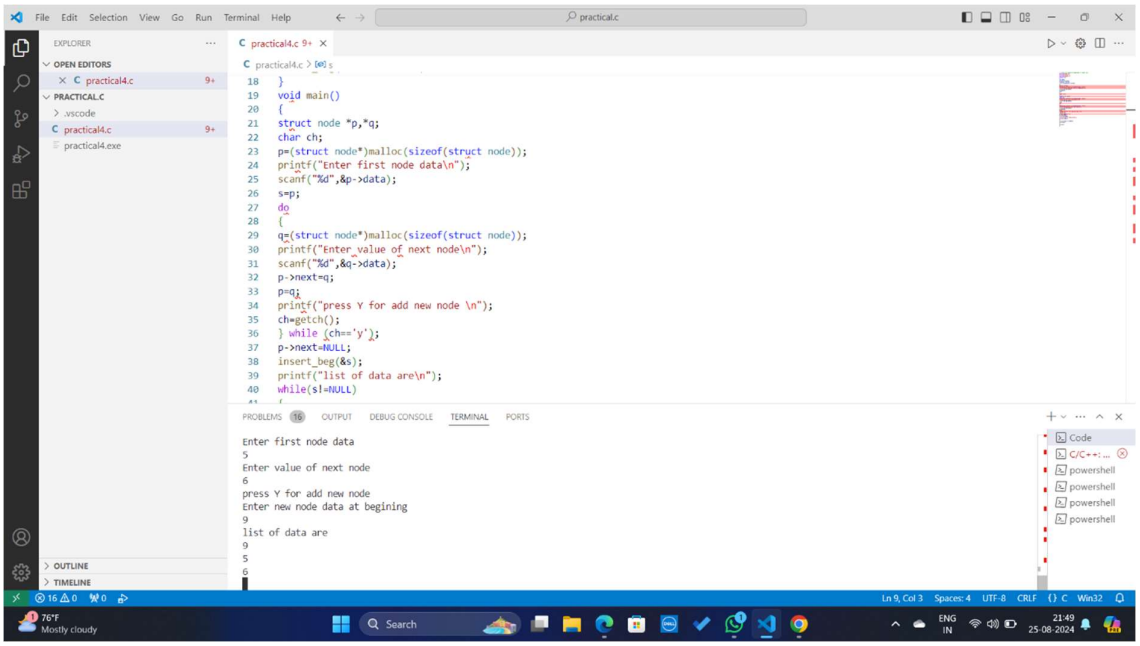
Program:

```
// Insert an element at beginning of Linked list

#include<stdio.h>

#include<malloc.h>

#include<stdlib.h>

struct node

{

int data;

struct node*next;

};struct node *s;

insert_beg(struct node**S)

{

struct node*p;
```

```
p=(struct node*)malloc(sizeof(struct node));
printf("Enter new node data at begining\n");
scanf("%d",&p->data);
p->next=s;
s=p;
}
void main()
{
struct node *p,*q;
char ch;
p=(struct node*)malloc(sizeof(struct node));
printf("Enter first node data\n");
scanf("%d",&p->data);
s=p;
do
{
q=(struct node*)malloc(sizeof(struct node));
printf("Enter value of next node\n");
scanf("%d",&q->data);
p->next=q;
p=q;
printf("press Y for add new node \n");
ch=getch();
} while (ch=='y');
p->next=NULL;
insert_beg(&s);
printf("list of data are\n");
while(s!=NULL)
{
printf("%d\n",s->data);
s=s->next;
```

```
}
getch();
}
```

Output:



Q. Write a Program to insert an element at a specific positon of linked list.

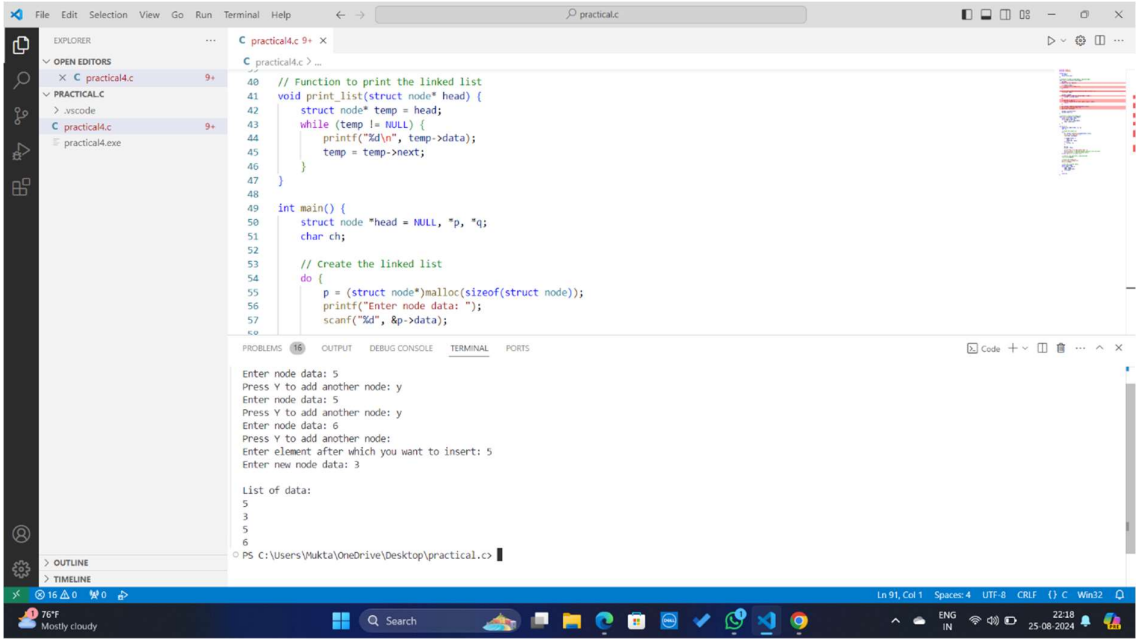
Program:

```
// insert element at specific postion of linked list
#include<stdio.h>
#include<malloc.h>
#include<stdlib.h>
struct node
{
int data;
struct node*next;
```

```
};struct node *s;
insert_after(struct node**S)
{
int key;
struct node*p,*q,*first;
printf("Enter element after which want to insert\n");
scanf("%d",&key);
first=s;
while(s->data!=key)
{
}
s=s->next;
p=(struct node*)malloc(sizeof(struct node));
printf("Enter new node data at any position\n");
   scanf("%d",&p->data);
   if(s->data==key)
   {
      q=s->next;
      s->next=p;
      p->next=q;
      s=first;
   }
   else{
      printf("not found\n");
   }
}

void main()
{
```

```
struct node *p,*q;
char ch;
p=(struct node*)malloc(sizeof(struct node));
printf("Enter first node data\n");
scanf("%d",&p->data);
s=p;
do
{
  q=(struct node*)malloc(sizeof(struct node));
  printf("Enter value of next node\n");
  scanf("%d",&q->data);
p->next=q;
p=q;
printf("press Y for add new node \n");
ch=getch();
} while (ch=='y');
p->next=NULL;
insert_after(&s);
printf("list of data are:\n");
while(s!=NULL)
{
}
printf("%d\n",s->data);
s=s->next;
getch();
}
```

Output:



Q. Write a Program to insert an element at end of the linked list.

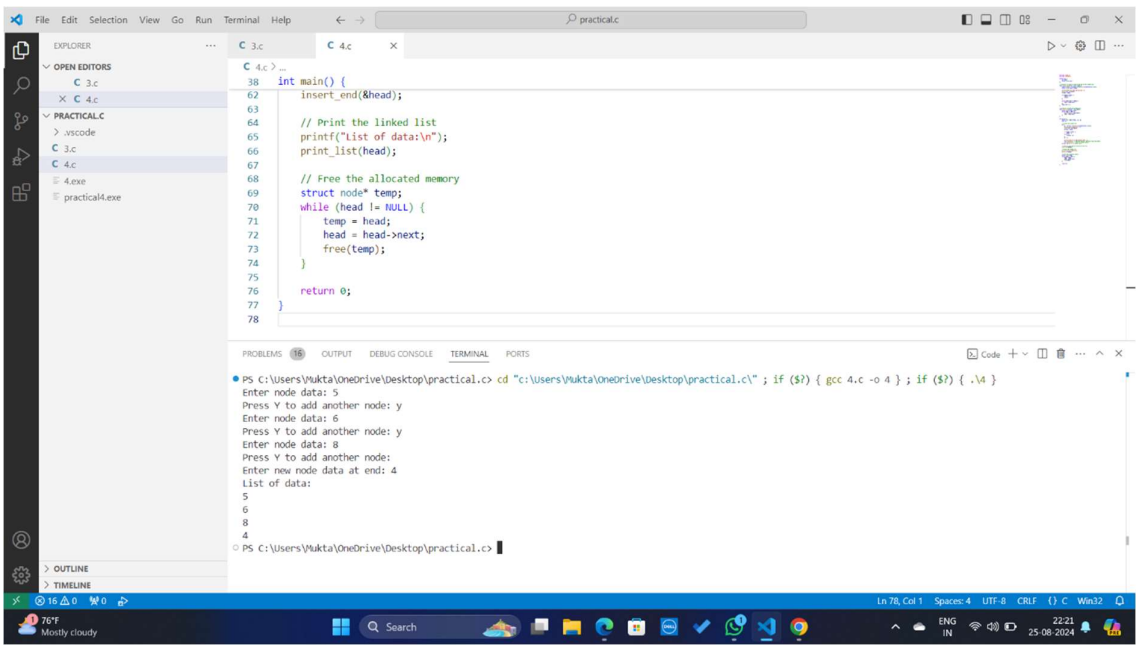
Program:

```
// insert at end of linked list
#include<stdio.h>
#include<malloc.h>
#include<stdlib.h>
struct node
{
int data;
struct node*next;
};struct node *s;
void insert_end(struct node**S)
{
struct node*t,*first;
t=(struct node*)malloc(sizeof(struct node));
printf("Enter new node data at End\n");
```

```
scanf("%d",&t->data);
first=s;
while(s->next!=NULL)
{
}
s=s->next;
s->next=t;
t->next=NULL;
s=first;
}
void main()
{
struct node *p,*q;
char ch;
p=(struct node*)malloc(sizeof(struct node));
printf("Enter first node data\n");
scanf("%d",&p->data);
s=p;
do
{
q=(struct node*)malloc(sizeof(struct node));
printf("Enter value of next node\n");
scanf("%d",&q->data);
p->next=q;
p=q;
printf("press Y for add new node \n");
ch=getchar();
ch=getchar();
} while (ch=='y');
```

```
p->next=NULL;
insert_end(&s);
printf("list of data are\n");
while(s!=NULL)
{
printf("%d\n",s->data);
s=s->next;
}
getchar();
}
```

Output:



Q . Write a program to delete an element at beginning of linked list.

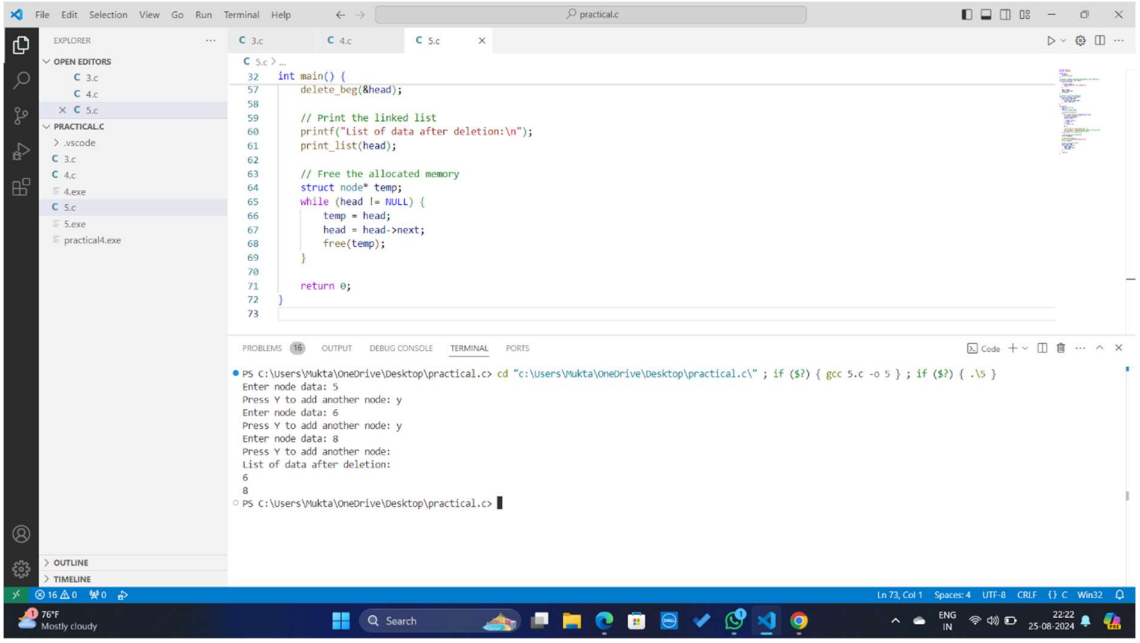
Program:

```
// delete from beg


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

```
#include<stdlib.h>

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

void delete_beg(struct node**S)
{
   struct node*p;

 if(s==NULL)
   {
      printf("Under flow");
      return ;
   }

   p=s;
   s=p->next;
free(p);
}
void main()
{
struct node *p,*q;
char ch;
p=(struct node*)malloc(sizeof(struct node));
printf("Enter first node data\n");
scanf("%d",&p->data);
```

```
s=p;
do
{
q=(struct node*)malloc(sizeof(struct node));
printf("Enter value of next node\n");
scanf("%d",&q->data);
p->next=q;
p=q;
printf("press Y for add new node \n");
ch=getch();
} while (ch=='y');
p->next=NULL;
delete_beg(&s);
printf("list of data are\n");
while(s!=NULL)
{
}
printf("%d\n",s->data);
s=s->next;
getch();
}
```

Output:



Q. Write a program delete specific element from linked list.

Program:

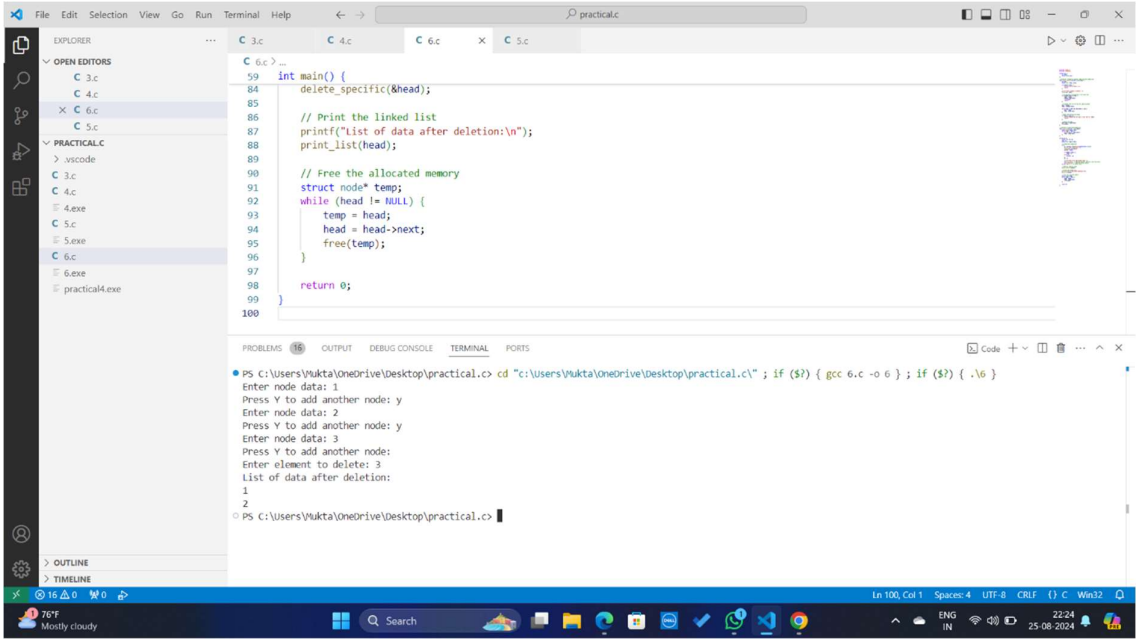
```
#include <stdio.h>

#include <stdlib.h>


struct node {

   int data;

   struct node* next;

};


// Function to delete a specific node from the linked list

void delete_specific(struct node** head) {

   int key;

   struct node *temp, *prev;


   if (*head == NULL) {
```

```
    printf("List is empty.\n");
    return;
}

printf("Enter element to delete: ");
scanf("%d", &key);

// If the node to be deleted is the head node
if ((*head)->data == key) {
    temp = *head;
    *head = temp->next;
    free(temp);
    return;
}

// Traverse the list to find the node to delete
prev = *head;
temp = (*head)->next;

while (temp != NULL && temp->data != key) {
    prev = temp;
    temp = temp->next;
}

// Node with the key not found
if (temp == NULL) {
    printf("Element %d not found in the list.\n", key);
    return;
}
```

```
    // Remove the node
    prev->next = temp->next;
    free(temp);
}

// Function to print the linked list
void print_list(struct node* head) {
    struct node* temp = head;
    while (temp != NULL) {
        printf("%d\n", temp->data);
        temp = temp->next;
    }
}

int main() {
    struct node *p, *q;
    char ch;
    struct node *head = NULL;

    // Create the linked list
    do {
        p = (struct node*)malloc(sizeof(struct node));
        printf("Enter node data: ");
        scanf("%d", &p->data);
        p->next = NULL;

        if (head == NULL) {
            head = p;
```

```
        } else {
            q->next = p;
        }
        q = p;

        printf("Press Y to add another node: ");
        ch = getchar(); // Read the newline character from the buffer
        ch = getchar(); // Read the actual choice
    } while (ch == 'Y' || ch == 'y');

    // Delete a specific node
    delete_specific(&head);

    // Print the linked list
    printf("List of data after deletion:\n");
    print_list(head);

    // Free the allocated memory
    struct node* temp;
    while (head != NULL) {
        temp = head;
        head = head->next;
        free(temp);
    }

    return 0;
}
```

Output:



Q. Write a program to delete an element at end of linked list.

Program:

```
#include <stdio.h>

#include <stdlib.h>


struct node {

   int data;

   struct node* next;

};


// Function to delete the last node from the linked list

void delete_end(struct node** head) {

   struct node *temp, *prev;


   if (*head == NULL) {

      printf("List is empty.\n");

      return;
```

```
    }

    if ((*head)->next == NULL) {
        // Only one node in the list
        free(*head);
        *head = NULL;
        return;
    }

    temp = *head;
    while (temp->next != NULL) {
        prev = temp;
        temp = temp->next;
    }

    // Now temp points to the last node
    prev->next = NULL;
    free(temp);
}

// Function to print the linked list
void print_list(struct node* head) {
    struct node* temp = head;
    while (temp != NULL) {
        printf("%d\n", temp->data);
        temp = temp->next;
    }
}
```

```
int main() {
    struct node *p, *q;
    char ch;
    struct node *head = NULL;

    // Create the linked list
    do {
        p = (struct node*)malloc(sizeof(struct node));
        printf("Enter node data: ");
        scanf("%d", &p->data);
        p->next = NULL;

        if (head == NULL) {
            head = p;
        } else {
            q->next = p;
        }
        q = p;

        printf("Press Y to add another node: ");
        ch = getchar(); // Read the newline character from the buffer
        ch = getchar(); // Read the actual choice
    } while (ch == 'Y' || ch == 'y');

    // Delete the last node
    delete_end(&head);

    // Print the linked list
    printf("List of data after deletion:\n");
```

```
    print_list(head);

    // Free the allocated memory
    struct node* temp;
    while (head != NULL) {
        temp = head;
        head = head->next;
        free(temp);
    }

    return 0;
}
```

Output: