Date
Page

```
void display()
{
    temp = head;
    printf ("The elements are: \n");
    while (temp != NULL) {
      printf ("%d\n", temp->data);
      temp= temp->next;
    }
}

void   delete_beg () {
         temp= head;
         if (head== NULL){
              printf ("List is empty"); }
         else {
              head= temp-> next;
              free(temp);
         }
}
void delete_end () {
         struct node* prenode;
         temp= head;
         while (temp-> next != NULL) {
              prenode= temp;
              temp= temp-> next; }
     if (temp == head) {
              head = NULL; }
     else {
              prenode-> next = 0; }
     free (temp);
}
```

```
void delete-pos()
{
        struct node* nextnode;
        int pos, i= 1;
        temp= head;
        printf ("enter position: \n");
        scanf ("%d", & pos, );
        while (i<pos) {
              temp= temp -> next;
              i++;
        }
        nextnode= temp-> next;
        temp-> next = nextnode ->next;
        free(nextnode);
}

void main () {
     int choice;
   while(1)
{  printf (enter operation: \n(1) create \n 2) display
              3) delete at beginning 4) delete at end \n
              5) delete at a position 6) to end \n");
   scanf ("%d", &choice );
    if(choice = = -1) {
         printf(" \n Operation completed")
      }
   else{
       switch(choice) {
        case 1: create ();
           break;
        case 2: display ();
          break;
```

(2) WAP to implement singly linked list with following operations:

(a) Create a linked list

(b) Deletion of first element, specified element & last element in the list.

(c) display the contents of the linked list.

```
⇒ # include <stdio.h>
# include <stdlib.h>
struct node {
          int data;
          struct node *next;
  };
struct node* head= NULL, *newnode, *temp;
void create () {
      int i, n;
      printf ("enter the no. of elements : \n");
      scanf ("%d", &n);
      for (i=0; i<n ; i++ ) {
        newnode= (struct node*) malloc (sizeof(struct node));
      printf ("enter the %d element :\n", i+1);
      scanf ("%d", & newnode ->data );
      newnode -> next = NULL;
      if (head==NULL ) {
           temp= head= newnode ;
      }
    else {
        temp-> next = newnode;
        temp= newnode;
    }
  }
}
```

```
case3: delete_beg();
    break;
case 4: delete_end();
    break;
case 5: delete_pos();
    break;
case 6: exit(0);
default: printf("invalid input");
    }
  }
}
}
```

22-1-23

LAB-4.

(1) WAP to implement singly linked list with following ops:

① Create a list

② Insertion of a node at first position, at any position & at the end of the list.

③ Display the contents of the linked list.

```
=> # include <stdio.h>
   # include <stdlib.h>
   struct node {
           int data;
         struct node* next;
   }
```

```
struct node* he
void create() {
      int i,n
   printf("
   scanf("
   for(i=0
      newno
      printf
      scanf
      newno
      if(he
      else {
```

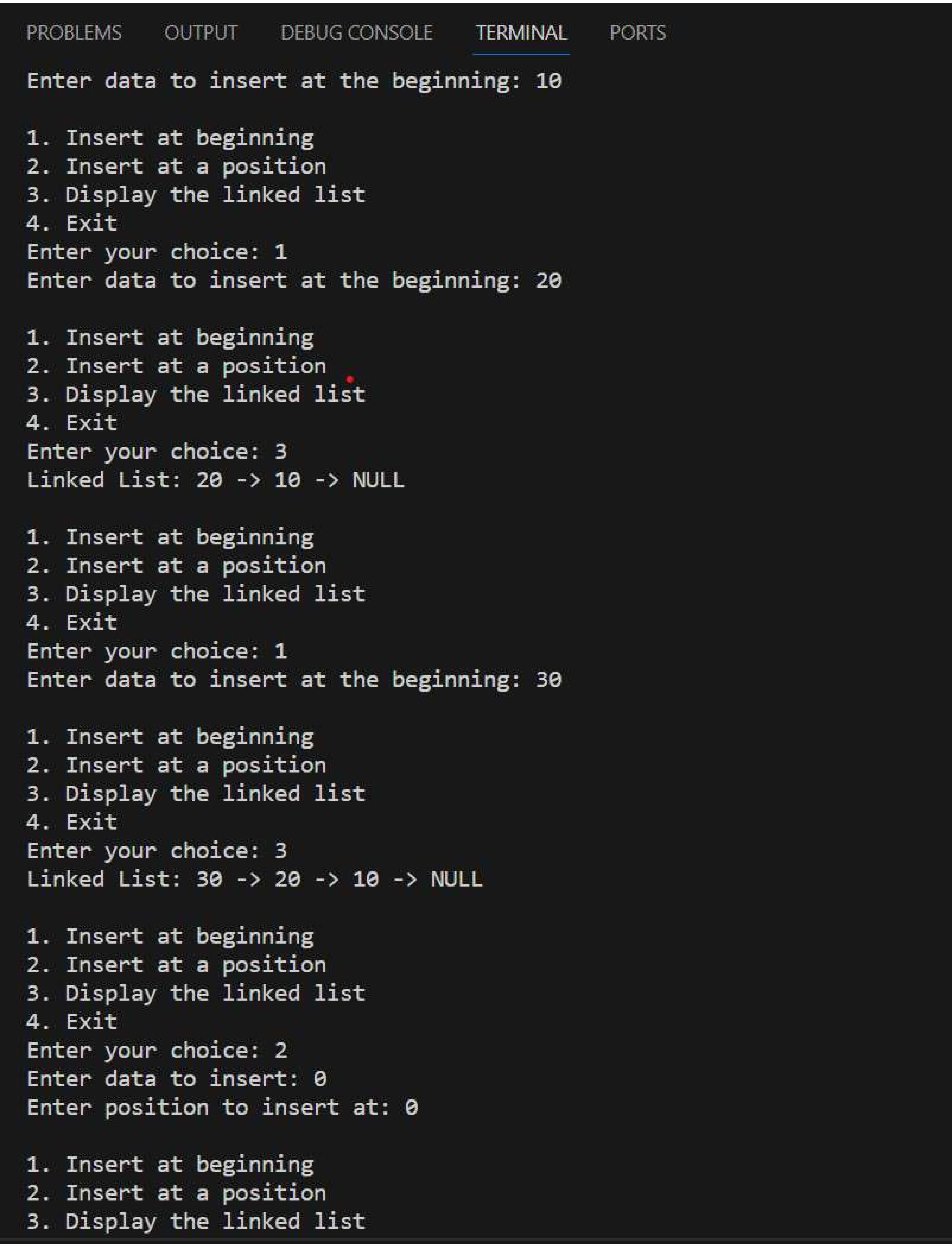
```
void insert_end()
{
    newnode= (struct node*)malloc (sizeof (struct no
    printf ("Enter the new element : \n");
    scanf ("%d", &newnode->data);
    newnode->next =NULL
    temp= head;
    while (temp -> next != NULL) {
        temp= temp->next;  }
    temp->next = newnode;

}
void insert_pos() {
    int pos, i=0;
    newnode (struct node* ) malloc (sizeof (struct no
    printf ("enter the position : \n");
    scanf ("%d", &pos );
    if (pos<0) {
        printf ("invalid pos". \n");  }
    else {
        temp= head;
        while (i< pos-1) {
            temp= temp-> next ;
            i++;  }
        printf ("enter the new element : \n");
        scanf ("%d", &newnode -> data);
        newnode -> next = temp-> next ;
        temp -> next = newnode ;
    }

void main() {
    int choice;
    while(1) {
```

Date
Page

```
printf ("enter operation\n 1) create , 2) display (3) insert at
        beginning \n 4) insert at end 5) insert at position
        6) to end \n");
scanf ("%d", &choice);
if (choice = = -1) {
    printf ("Operation completed \n");
    break;
  }
  else {
    switch (choice)
      {
        case 1: create ();
        break;
        case 2: display ();
        break;
        case 3: insert_beg ();
        break;
        case 4: insert_end ();
        break;
        case 5: insert_pos ();
        break;
        case 6: exit (0);
        default : printf ("invalid input ");
   }
  }
 }
}
```

24/1/24

```
struct node* ~~ae~~head = NULL, *newnode, *temp;
void create() {
     int i,n;
    printf(" Enter the no of elements: ");
    scanf("%d", &n);
    for(i=0; i<n; i++) {
      newnode= (struct node* ) malloc(sizeof(struct node));
      printf("enter the %d element\n", i+1);
      scanf("%d", & newnode -> data);
      newnode ->next = NULL;
      if(head = =0) {
              temp =head= newnode; }
     else {
           temp -> next = newnode;
           temp= newnode;
        }
    }
}
void display() {
        temp= head;
        printf(" The elemente are: \n");
       while ( temp! = NULL) {
         printf ("%d\n", temp->data );
         temp= temp -> next;
}
}
void insert_beg() {
   newnode= (struct node*) malloc(sizeof(struct node
   printf ("enter the new element: ");
   scanf ("%d", & newnode ->data );
  newnode -> next = head;
   head= newnode;
}
```



```
Enter data to insert at the beginning: 10

1. Insert at beginning
2. Insert at a position
3. Display the linked list
4. Exit
Enter your choice: 1
Enter data to insert at the beginning: 20

1. Insert at beginning
2. Insert at a position
3. Display the linked list
4. Exit
Enter your choice: 3
Linked List: 20 -> 10 -> NULL

1. Insert at beginning
2. Insert at a position
3. Display the linked list
4. Exit
Enter your choice: 1
Enter data to insert at the beginning: 30

1. Insert at beginning
2. Insert at a position
3. Display the linked list
4. Exit
Enter your choice: 3
Linked List: 30 -> 20 -> 10 -> NULL

1. Insert at beginning
2. Insert at a position
3. Display the linked list
4. Exit
Enter your choice: 2
Enter data to insert: 0
Enter position to insert at: 0

1. Insert at beginning
2. Insert at a position
3. Display the linked list
```