LAB PROGRAM-7

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

 struct node
{
 int info;
 struct node *link;
};
 typedef struct node *NODE;

 NODE getnode()
{
 NODE x;
 x = (NODE)malloc(sizeof(struct node));
 if(x==NULL)
{
printf("\n Memory is full\n");
 exit(0);
}
 return x;
}
NODE insert_front(NODE first, int item)
{
 NODE temp;
temp = getnode();
temp->info = item;
temp->link = NULL;
 if(first==NULL)
{
 return temp;
}
```

Scanned with CamScanner

```
temp -> link = first;
first = temp;
return first;
}
NODE delete_front (NODE first)
{
 NODE temp;
 if (first == NULL)
{
 printf("List is empty. Cannot delete\n");
      return first;
}
 temp = first;
 temp = temp -> link;
 printf("Item deleted at front end is %d\n", first -> info);
  free (first);
  return temp;
}
NODE IF (NODE second, int item)
{
 NODE temp;
 temp = getnode ();
 temp -> info = item;
 temp -> link = NULL;
 if (second == NULL)
  return temp;
 temp -> link = second;
  second = temp;
  return second;
}
NODE IR (NODE second, int item)
{
```

Scanned with CamScanner

```
NODE temp, cur ;
 temp = getnode ( ) ;
 temp -> info = item ;
 temp -> link = NULL ;
 if( second = = NULL)
   return temp ;
 cur = second ;
 while(cur -> link! = NULL)
      cur = cur -> link ;
      cur -> link = temp ;
  return second ;
}
 NODE reverse (NODE first)
{
  NODE cur, temp ;
   cur = NULL ;
  while (first ! = NULL)
{
  temp = first ;
  first = first -> link ;
   temp -> link = cur ;
     cur = temp ;
}
  return cur ;
}
  NODE ascending (NODE first)
{
   NODE prev = first ;
   NODE cur = NULL ;
     int temp ;
   if(first== NULL)
{
```

Scanned with CamScanner

```
return 0;
}
else
{
  while (prev! = NULL)
{
  cur = prev -> link;
  while(cur! = NULL)
{
  if (prev -> info > cur -> info)
{
  temp = prev -> info;
  prev -> info = cur -> info;
  cur -> info = temp;
}
  cur = cur -> link;
}
  prev = prev -> link;
}
}
  return first;
}
NODE descending (NODE first)
{
  NODE prev = first;
  NODE cur = NULL;
  int temp;
  if(first == NULL)
{
  return 0;
}
  else
```

Scanned with CamScanner

```
return first ;
}
void display (NODE first)
{
NODE temp ;
if(first == NULL)
 printf("List is empty . Cannot display items . \n");
 printf("List contents are :");
 for(temp = first ; temp ! = NULL ; temp = temp -> link)
 {
 printf("\n%d", temp -> info) ;
 }
}
 void main.()
 {
  int
item , choice , pos , element , option , choice2 , item I , num ;
   NODE first = NULL ;
   NODE second = NULL ;
   for (;;)
{
 printf ("\n\n Choose an option") ;
 printf ("\n1 : Insert_front \n2 : Delete front \n3 : Reverse
        \n4 : Sort \n5 : Concatenate \n6 : Display \n : Exit\n);
  printf ("Enter the choice :") ;
   scanf ("%d", &choice) ;
   switch (choice)
 {
 case 1 : printf ("Enter the item at front-end :") ;
          scanf ("%d", &item) ;
first = insert_front (first, item) ;
        printf ("%d inserted at front-end .", first -> info) ;
```

Scanned with CamScanner

```
break;
case 2:
first = delete_front(first);
    break;
case 3: first = reverse(first);
printf("List is reversed.");
break;
case 4: printf("Press 1 for Ascending-sort and 2 for
                Descending-sort:");
        scanf("%d", &option);
    if (option == 1)
{
  first = ascending(first);
printf("List is sorted in ascending order.");
}
  if (option == 2)
{
  first = descending(first);
  printf("List is sorted in descending order.");
}
  break;
case 5: printf("Create a second list\n");
printf("Enter the no. of elements in the second list:");
  scanf("%d", &num);
  for(int i = 1; i <= num; i++)
{
  printf("\n Press 1 to Insert-front & 2 to Insert-rear:");
    scanf("%d", &choice2);
  if (choice2 == 1)
  {
  printf("Enter the item at front-end:");
    scanf("%d", &item1);
```

Scanned with CamScanner

```
second = IF (second, item1);
  }
   if (choice 2 == 2)
  {
   printf ("Enter the item at rear-end : ");
    scanF ("%d", &item1);

second IR = (second, item1);
   }
 }
first= concatenate (first, second);
    printf("\n The two lists are concatenated.");
      break;
 case 6 : display (first);
    break;
  default : exit(0);
    break;
 }
}
}
```

Scanned with CamScanner

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

struct node
{
    int info;
    struct node *link;
};
typedef struct node *NODE;

NODE getnode()
{
    NODE x;
    x = (NODE)malloc(sizeof(struct node));
    if(x==NULL)
    {
        printf("\nMemory is full\n");
        exit(0);
    }
    return x;
}

NODE insert_front(NODE first,int item)
{
    NODE temp;
    temp=getnode();
    temp->info=item;
    temp->link=NULL;
    if(first==NULL)
```

Scanned with CamScanner

main.c

```
29          if(first==NULL)
30          {
31              return temp;
32          }
33          temp->link=first;
34          first=temp;
35          return first;
36      }
37
38      NODE delete_front(NODE first)
39      {
40          NODE temp;
41          if(first==NULL)
42          {
43              printf("List is empty. Cannot delete\n");
44              return first;
45          }
46          temp=first;
47          temp = temp->link;
48          printf("Item deleted at front end is %d\n",first->info);
49          free(first);
50          return temp;
51      }
52
53      NODE IF(NODE second,int item)
54      {
55          NODE temp;
56          temp=getnode();
57          temp->info=item;
```

Scanned with CamScanner

```
    temp->link=NULL;
    if(second==NULL)
        return temp;
    temp->link=second;
    second=temp;
    return second;
}

NODE IR(NODE second,int item)
{
    NODE temp,cur;
    temp=getnode();
    temp->info=item;
    temp->link=NULL;
    if(second==NULL)
        return temp;
    cur=second;
    while(cur->link!=NULL)
        cur=cur->link;
    cur->link=temp;
    return second;
}

NODE reverse(NODE first)
{
    NODE cur,temp;
    cur=NULL;
    while(first!=NULL)
```

Scanned with CamScanner

main.c

```
86          {
87              temp=first;
88              first=first->link;
89              temp->link=cur;
90              cur=temp;
91          }
92          return cur;
93      }
94
95      NODE ascending(NODE first)
96      {
97          NODE prev=first;
98          NODE cur=NULL;
99          int temp;
100         if(first== NULL)
101         {
102             return 0;
103         }
104             else
105             {
106               while(prev!= NULL)
107               {
108                   cur = prev->link;
109                   while(cur!= NULL)
110                   {
111                           if(prev->info > cur->info)
112                           {
113                             temp = prev->info;
114                             prev->info = cur->info;
```

Scanned with CamScanner

ain.c

```
                prev->info = cur->info;
                cur->info = temp;
            }
            cur = cur->link;
        }
        prev= prev->link;
    }
  }
  return first;
}

NODE concatenate(NODE first,NODE second)
{
    NODE cur;
    if(first==NULL)
        return second;
    if(second==NULL)
        return first;
    cur=first;
    while(cur->link!=NULL)
    {
        cur=cur->link;
    }
    cur->link=second;
    return first;
}

void display(NODE first)
{
```

Scanned with CamScanner

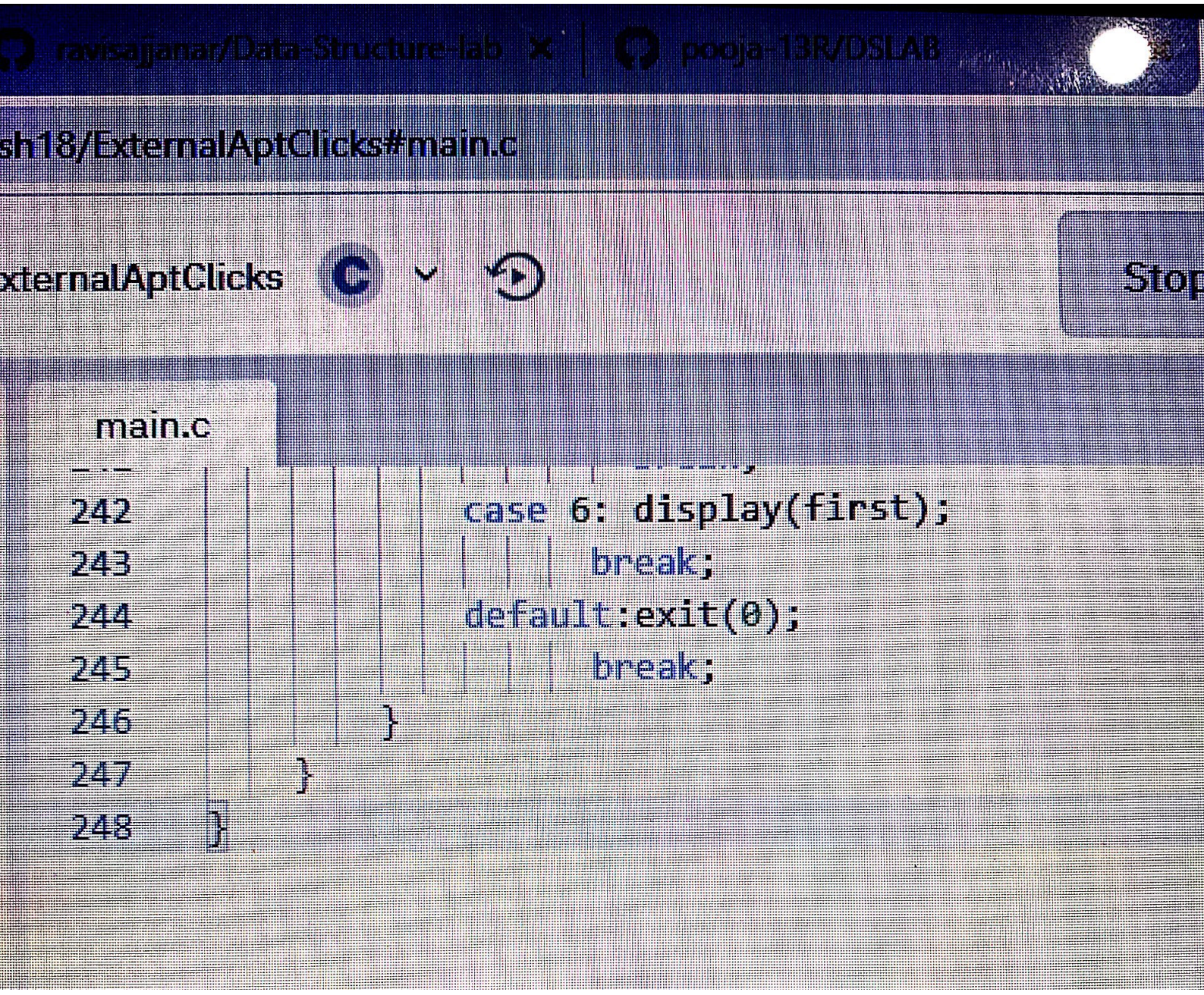
```
            scanf("%d",&item);
            first=insert_front(first,item);
            printf("%d inserted at front-end.",
            first->info);
            break;
        case 2: first=delete_front(first);
            break;
        case 3: first=reverse(first);
            printf("List is reversed.");
            break;
        case 4: printf("Press 1 for Ascending-sort and
        2 for Descending-sort : ");
            scanf("%d",&option);
            if(option==1)
            {
                first=ascending(first);
                printf("List is sorted in ascending
                order.");
            }
            if(option==2)
            {
                first=descending(first);
                printf("List is sorted in
                descending order.");
            }
            break;
        case 5: printf("Create a second list\n");
            printf("Enter the number of elements in
            the second list : ");
```

Scanned with CamScanner

main.c

```
219             case 5: printf("Create a second list\n");
220                     printf("Enter the number of elements in
                        the second list : ");
221                     scanf("%d",&num);
222                     for(int i=1;i<=num;i++)
223                     {
224                         printf("\nPress 1 to Insert-front
                            and 2 to Insert-rear : ");
225                         scanf("%d",&choice2);
226                         if(choice2==1)
227                         {
228                             printf("Enter the item at
                                front-end : ");
229                             scanf("%d",&item1);
230                             second=IF(second,item1);
231                         }
232                         if(choice2==2)
233                         {
234                             printf("Enter the item at
                                rear-end : ");
235                             scanf("%d",&item1);
236                             second=IR(second,item1);
237                         }
238                     }
239                     first=concatenate(first,second);
240                     printf("\nThe two lists are
                        concatenated.");
241                     break;
242             case 6: display(first);
```

Scanned with CamScanner



Scanned with CamScanner

```
$ clang-7 -pthread -lm -o main main.c
main.c:183:1: warning: return type of 'main' is not 'int'
in-return-type]
void main()
^
main.c:183:1: note: change return type to 'int'
void main()
^~~~
int
1 warning generated.
$ ./main


Choose an option
1:Insert_front
2:Delete_front
3:Reverse
4:Sort
5.Concatenate
6:Display
7:Exit
Enter the choice : 1
Enter the item at front-end : 23
23 inserted at front-end.

Choose an option
1:Insert_front
2:Delete_front
3:Reverse
4:Sort
5.Concatenate
6:Display
7:Exit
Enter the choice : 1
Enter the item at front-end : 34
```

Scanned with CamScanner

```
Enter the item at front-end : 34
34 inserted at front-end.

Choose an option
1:Insert_front
2:Delete_front
3:Reverse
4:Sort
5.Concatenate
6:Display
7:Exit
Enter the choice : 3
List is reversed.

Choose an option
1:Insert_front
2:Delete_front
3:Reverse
4:Sort
5.Concatenate
6:Display
7:Exit
Enter the choice : 5
Create a second list
Enter the number of elements in the second list : 45

Press 1 to Insert-front and 2 to Insert-rear : 1
Enter the item at front-end : 32

Press 1 to Insert-front and 2 to Insert-rear : 2
Enter the item at rear-end : 44

Press 1 to Insert-front and 2 to Insert-rear : 6

Press 1 to Insert-front and 2 to Insert-rear :
```

Scanned with CamScanner