LAB-8:--

PROGRAM

```
#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("mem full\n");
```

```
    exit(0);
  }
  return x;
}
void freenode(NODE x)
{
free(x);
}
NODE insert_rear(NODE first,int item)
{
NODE temp,cur;
temp=getnode();
temp->info=item;
temp->link=NULL;
if(first==NULL)
  return temp;
cur=first;
```

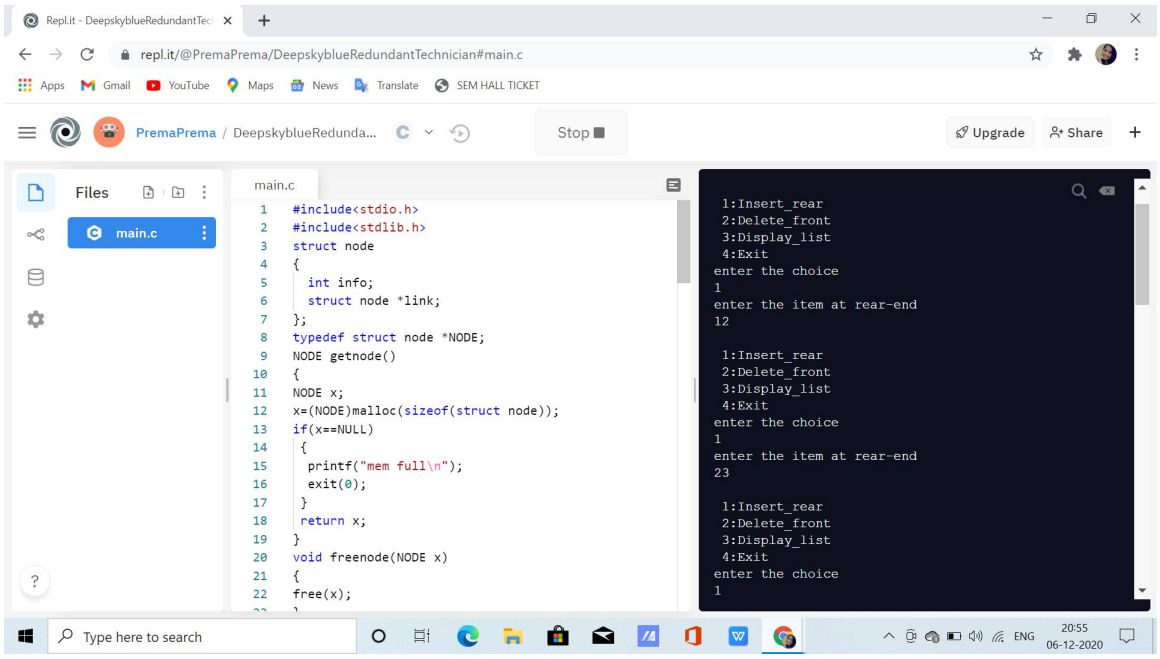
```
while(cur->link!=NULL)
  cur=cur->link;
cur->link=temp;
return first;
}

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);
```

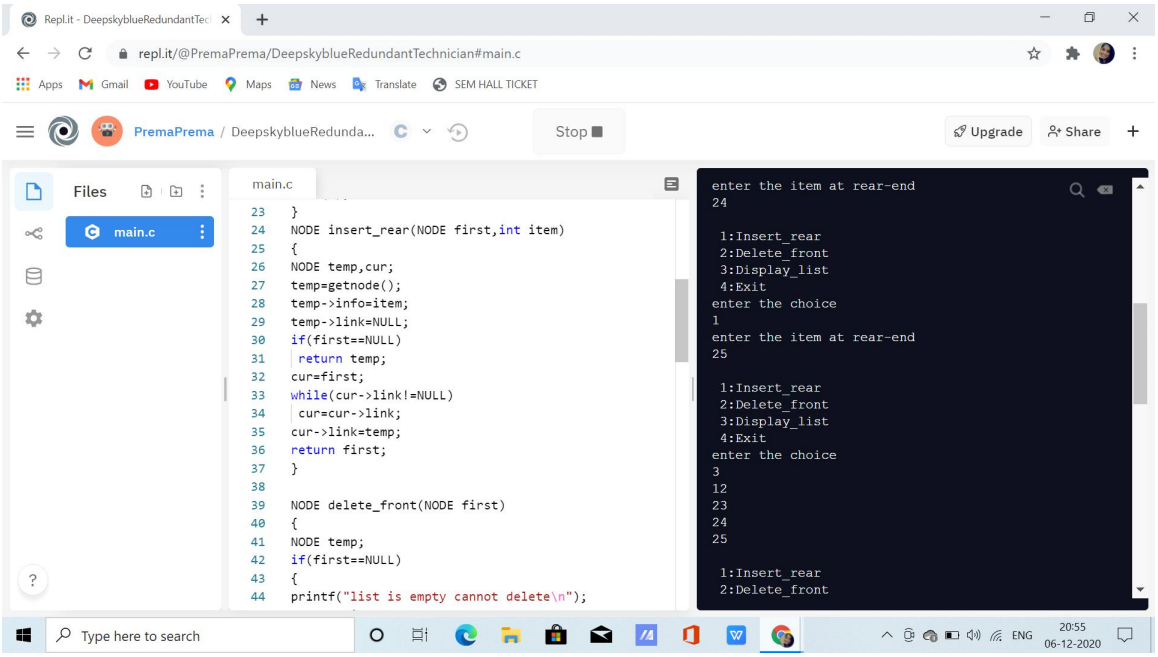
```
free(first);
return temp;
}
void display(NODE first)
{
  NODE temp;
  if(first==NULL)
  printf("list empty cannot display items\n");
  for(temp=first;temp!=NULL;temp=temp->link)
    {
    printf("%d\n",temp->info);
    }
}
int main()
{
int item,choice,pos;
NODE first=NULL;
```

```
for(;;)
{
printf("\n 1:Insert_rear\n 2:Delete_front\n 3:Display_list\n 4:Exit\n");
printf("enter the choice\n");
scanf("%d",&choice);
switch(choice)
 {
   case 1:printf("enter the item at rear-end\n");
          scanf("%d",&item);
          first=insert_rear(first,item);
          break;
   case 2:first=delete_front(first);
          break;
   case 3:display(first);
          break;
 default:exit(0);
```

```
        break;

    }

}

}
```

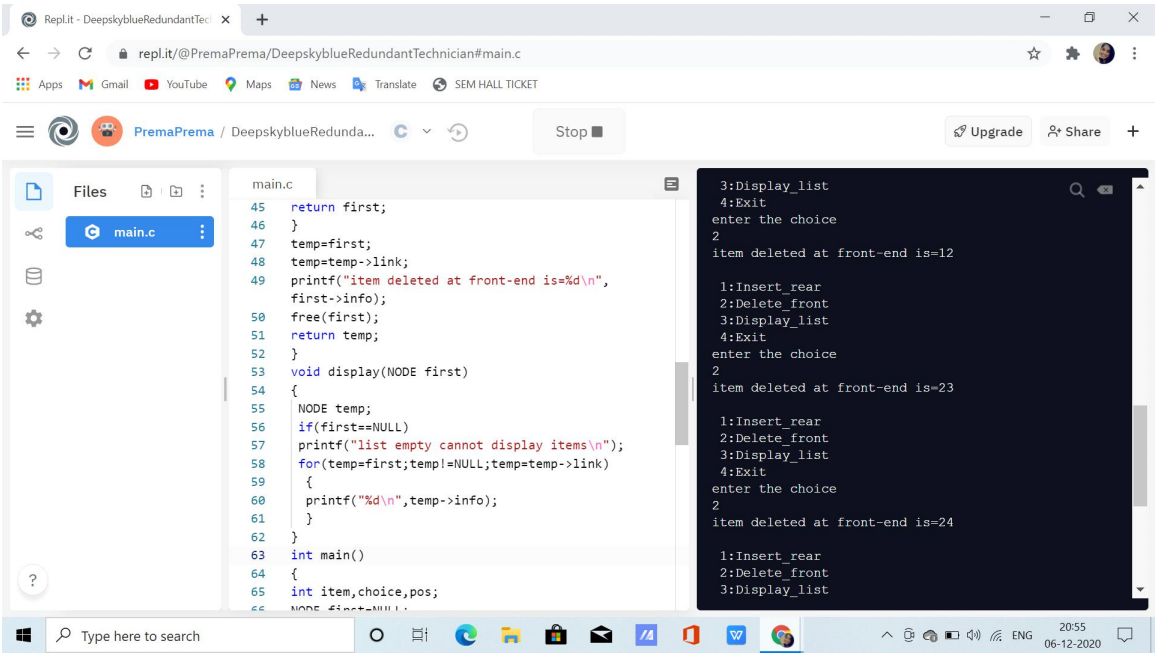
SCREENSHOT:--





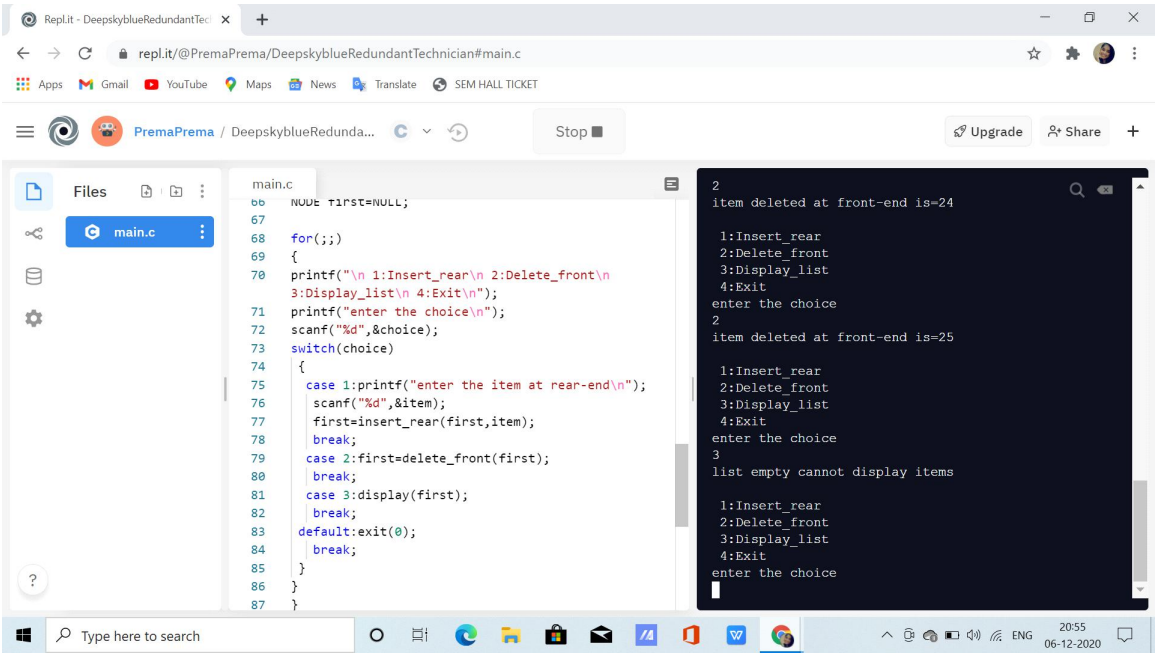
```
}
NODE insert_rear(NODE first,int item)
{
NODE temp,cur;
temp=getnode();
temp->info=item;
temp->link=NULL;
if(first==NULL)
 return temp;
cur=first;
while(cur->link!=NULL)
 cur=cur->link;
cur->link=temp;
return first;
}

NODE delete_front(NODE first)
{
NODE temp;
if(first==NULL)
{
printf("list is empty cannot delete\n");
```

```
enter the item at rear-end
24

 1:Insert_rear
 2:Delete_front
 3:Display_list
 4:Exit
enter the choice
1
enter the item at rear-end
25

 1:Insert_rear
 2:Delete_front
 3:Display_list
 4:Exit
enter the choice
3
12
23
24
25

 1:Insert_rear
 2:Delete_front
```



```
return first;
}
temp=first;
temp=temp->link;
printf("item deleted at front-end is=%d\n",
first->info);
free(first);
return temp;
}
void display(NODE first)
{
 NODE temp;
 if(first==NULL)
 printf("list empty cannot display items\n");
 for(temp=first;temp!=NULL;temp=temp->link)
  {
  printf("%d\n",temp->info);
  }
}
int main()
{
int item,choice,pos;
```

```
 3:Display_list
 4:Exit
enter the choice
2
item deleted at front-end is=12

 1:Insert_rear
 2:Delete_front
 3:Display_list
 4:Exit
enter the choice
2
item deleted at front-end is=23

 1:Insert_rear
 2:Delete_front
 3:Display_list
 4:Exit
enter the choice
2
item deleted at front-end is=24

 1:Insert_rear
 2:Delete_front
 3:Display_list
```



WRITTEN:---

IMPLEMENTING STACKS:--

program:--

#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("mem full\n");
    exit(0);
  }
  return x;
}
void freenode(NODE x)
```

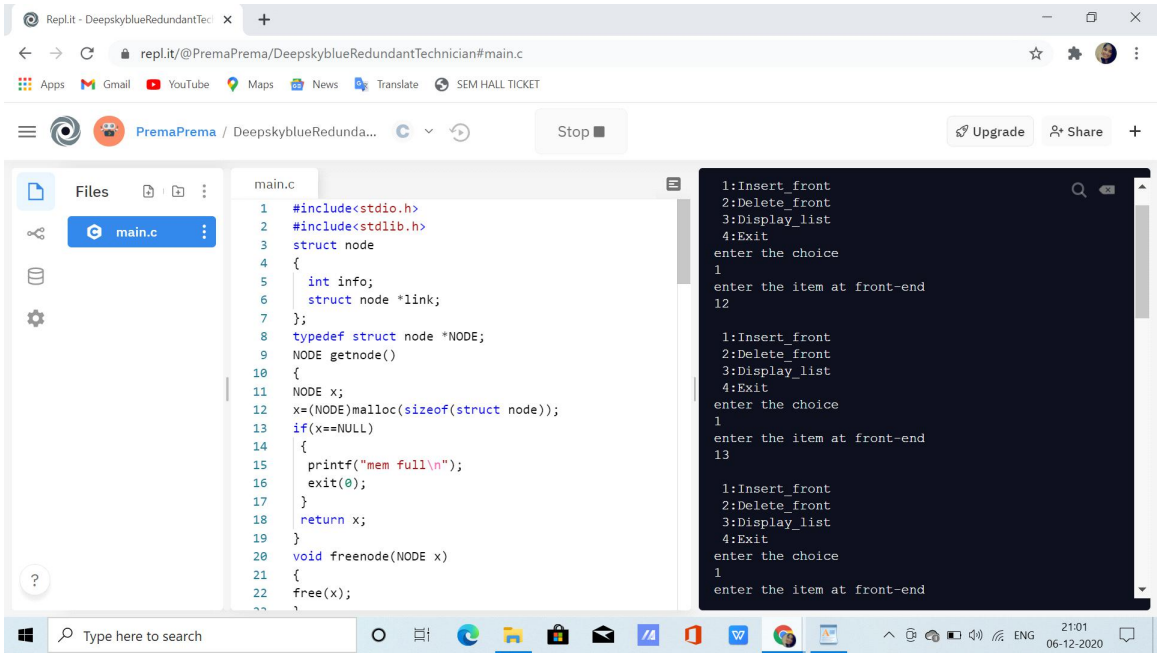
```
{
free(x);
}
NODE insert_front(NODE first,int item)
{
NODE temp;
temp=getnode();
temp->info=item;
temp->link=NULL;
if(first==NULL)
return temp;
temp->link=first;
first=temp;
return first;
}
NODE delete_front(NODE first)
{
```

```
NODE temp;
if(first==NULL)
{
printf("stack 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;
}
void display(NODE first)
{
  NODE temp;
  if(first==NULL)
  printf("stack empty cannot display items\n");
```

```
  for(temp=first;temp!=NULL;temp=temp->link)
    {
    printf("%d\n",temp->info);
    }
}
int main()
{
int item,choice,pos;
NODE first=NULL;


for(;;)
{
printf("\n 1:Insert_front\n 2:Delete_front\n 3:Display_list\n 4:Exit\n");
printf("enter the choice\n");
scanf("%d",&choice);
switch(choice)
  {
```

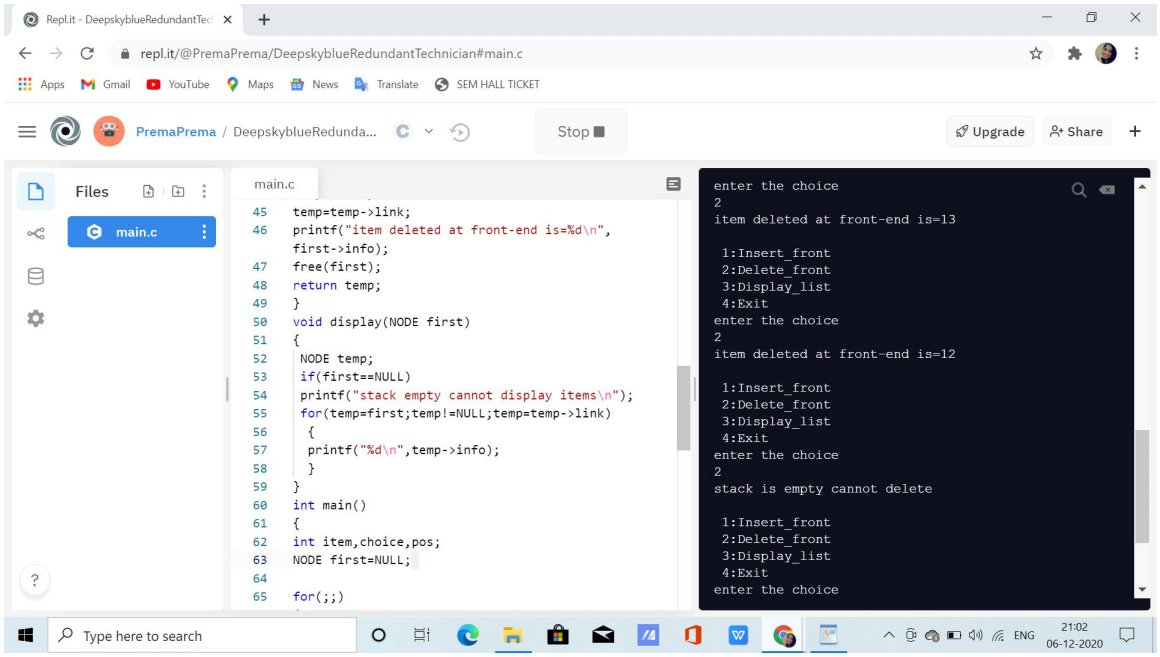
```
    case 1:printf("enter the item at front-end\n");
           scanf("%d",&item);
           first=insert_front(first,item);
           break;
    case 2:first=delete_front(first);
           break;
    case 3:display(first);
           break;
  default:exit(0);
           break;
   }
}
}
```

screenshots:--



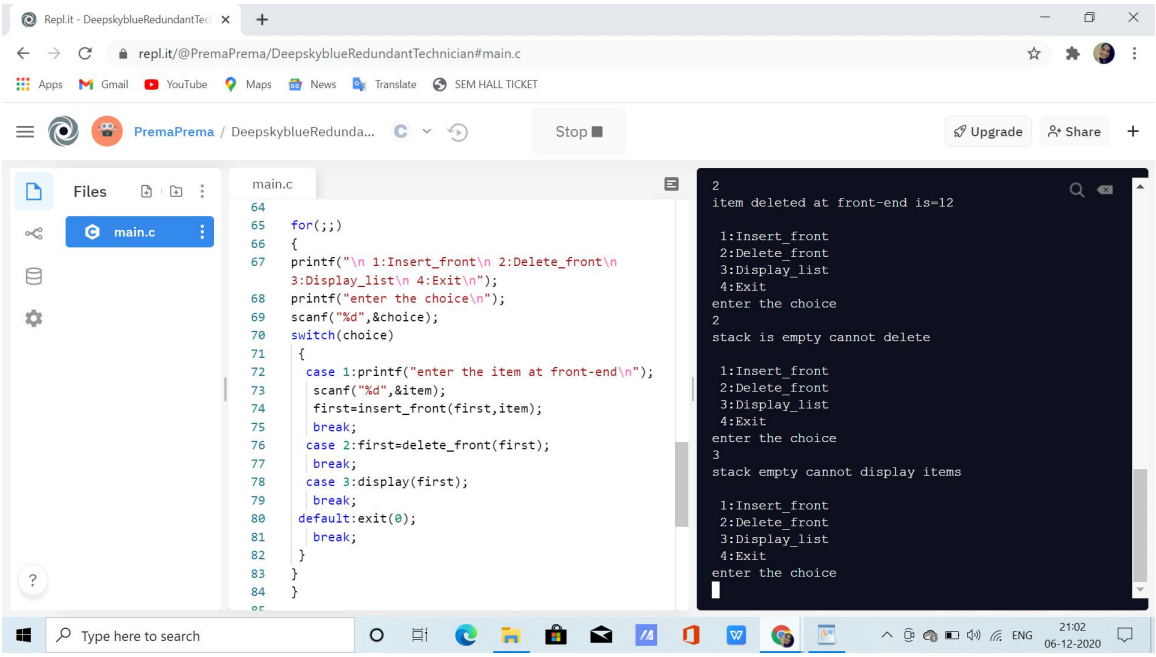
```
#include<stdio.h>
#include<stdlib.h>
struct node
{
  int info;
  struct node *link;
};
typedef struct node *NODE;
NODE getnode()
{
NODE x;
x=(NODE)malloc(sizeof(struct node));
if(x==NULL)
 {
  printf("mem full\n");
  exit(0);
 }
 return x;
}
void freenode(NODE x)
{
free(x);
```

```
 1:Insert_front
 2:Delete_front
 3:Display_list
 4:Exit
enter the choice
1
enter the item at front-end
12

 1:Insert_front
 2:Delete_front
 3:Display_list
 4:Exit
enter the choice
1
enter the item at front-end
13

 1:Insert_front
 2:Delete_front
 3:Display_list
 4:Exit
enter the choice
1
enter the item at front-end
```



```
  temp=temp->link;
  printf("item deleted at front-end is=%d\n",
  first->info);
  free(first);
  return temp;
  }
  void display(NODE first)
  {
   NODE temp;
   if(first==NULL)
   printf("stack empty cannot display items\n");
   for(temp=first;temp!=NULL;temp=temp->link)
    {
    printf("%d\n",temp->info);
    }
  }
  int main()
  {
  int item,choice,pos;
  NODE first=NULL;

  for(;;)
```

```
enter the choice
2
item deleted at front-end is=13

 1:Insert_front
 2:Delete_front
 3:Display_list
 4:Exit
enter the choice
2
item deleted at front-end is=12

 1:Insert_front
 2:Delete_front
 3:Display_list
 4:Exit
enter the choice
2
stack is empty cannot delete

 1:Insert_front
 2:Delete_front
 3:Display_list
 4:Exit
enter the choice
```


Repl.it - DeepskyblueRedundantTec
repl.it/@PremaPrema/DeepskyblueRedundantTechnician#main.c
Apps Gmail YouTube Maps News Translate SEM HALL TICKET
PremaPrema / DeepskyblueRedunda...
Stop
Upgrade
Share
Files
main.c
main.c
64
65 for(;;)
66 {
67 printf("\n 1:Insert_front\n 2:Delete_front\n
3:Display_list\n 4:Exit\n");
68 printf("enter the choice\n");
69 scanf("%d",&choice);
70 switch(choice)
71 {
72 case 1:printf("enter the item at front-end\n");
73 scanf("%d",&item);
74 first=insert_front(first,item);
75 break;
76 case 2:first=delete_front(first);
77 break;
78 case 3:display(first);
79 break;
80 default:exit(0);
81 break;
82 }
83 }
84 }
2
item deleted at front-end is=12
1:Insert_front
2:Delete_front
3:Display_list
4:Exit
enter the choice
2
stack is empty cannot delete
1:Insert_front
2:Delete_front
3:Display_list
4:Exit
enter the choice
3
stack empty cannot display items
1:Insert_front
2:Delete_front
3:Display_list
4:Exit
enter the choice
Type here to search
ENG
21:02
06-12-2020

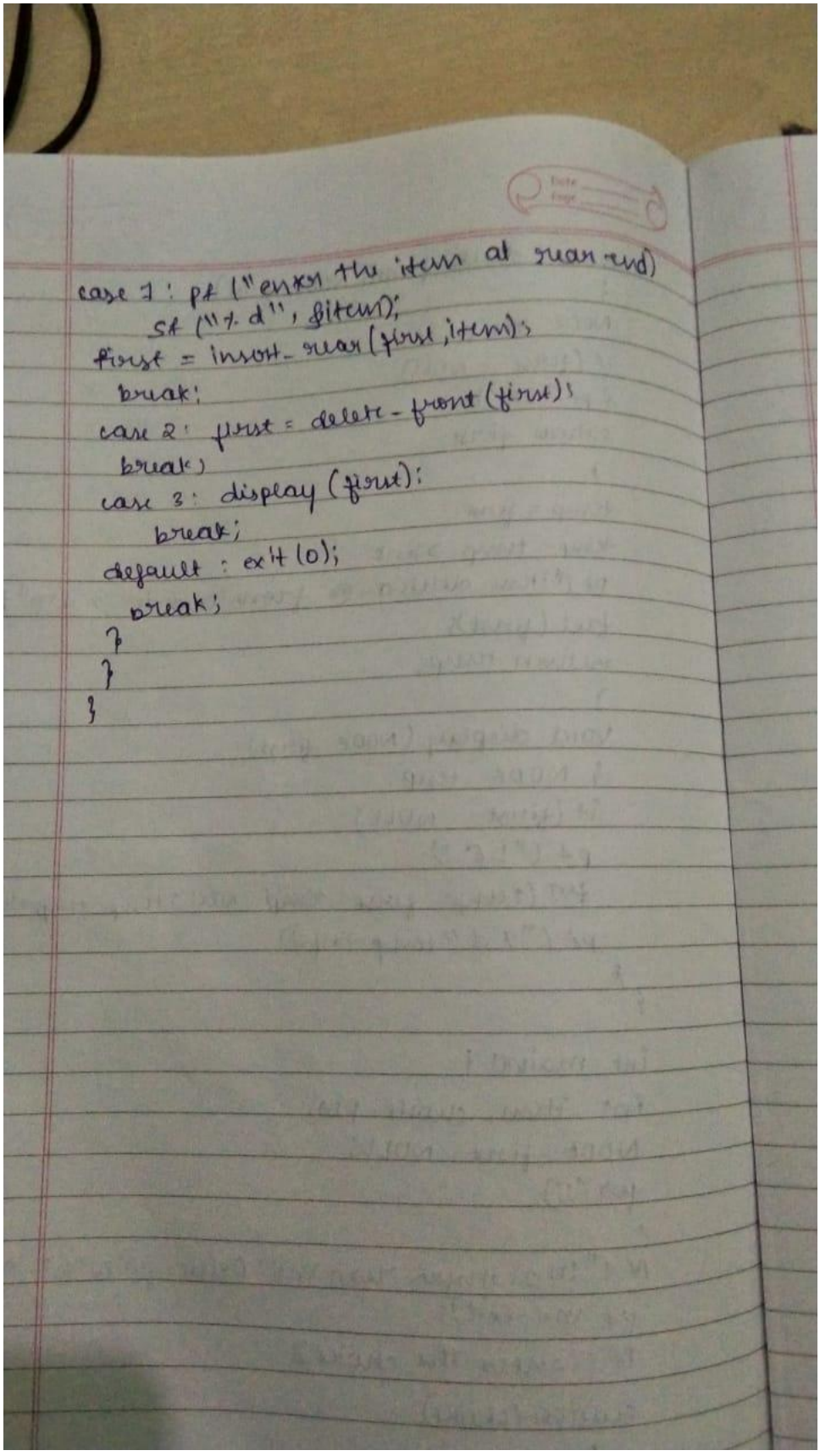
WRITTEN:

Lab 8

① Stack implementation.

```
#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("mem full\n");
exit(0); }
return x; }
void freenode(NODE x)
 {
 free(x); }
NODE insert_rear( NODE first, int item) {
NODE temp, cur;
temp = getnode();
temp -> info = item;
 temp -> link = NULL;
if (first == NULL) return temp;
cur = first;
while (cur -> link != NULL)
 cur = cur -> link
 cur -> link = temp return first;
}
```

```
NODE delete_front(NODE first)
{
NODE temp;
if (first == NULL)
{ pf("LE ");
return first;
 }
temp = first
temp = temp -> link;
pf("item deleted @ front end is = %d\n" first->
free(first);
return temp;
 }
void display (NODE first)
{ NODE temp;
 if (first == NULL)
  pf ("LE ");
  for (temp= first; temp != NULL; temp= temp->link){
  pf ("%d " temp->info);
}
int main() {
int item, choice, pos;
NODE first = NULL;
for (;;)
{
pf ("\n 1. insert_rear \n2: Delete_from \n3: display
\n4 \n4 Exit);
pf ("enter the choice");
switch (choice)
  {
```



```
case 1: pf ("enter the item at rear end)
    sf ("%d", &item);
first = insert_rear (first, item);
 break;
case 2: first = delete_front (first);
 break;
case 3: display (first);
   break;
default : exit (0);
 break;
 }
 }
}
```