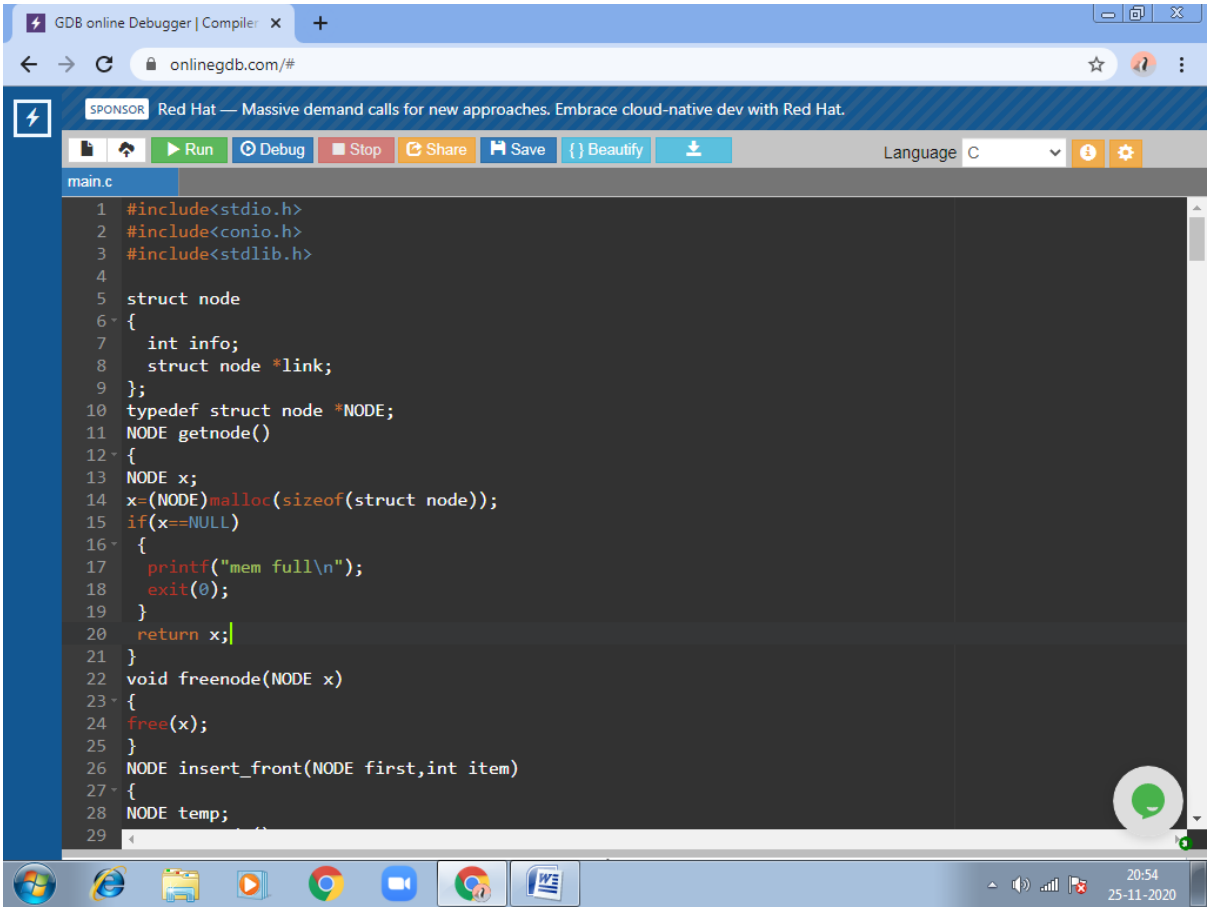
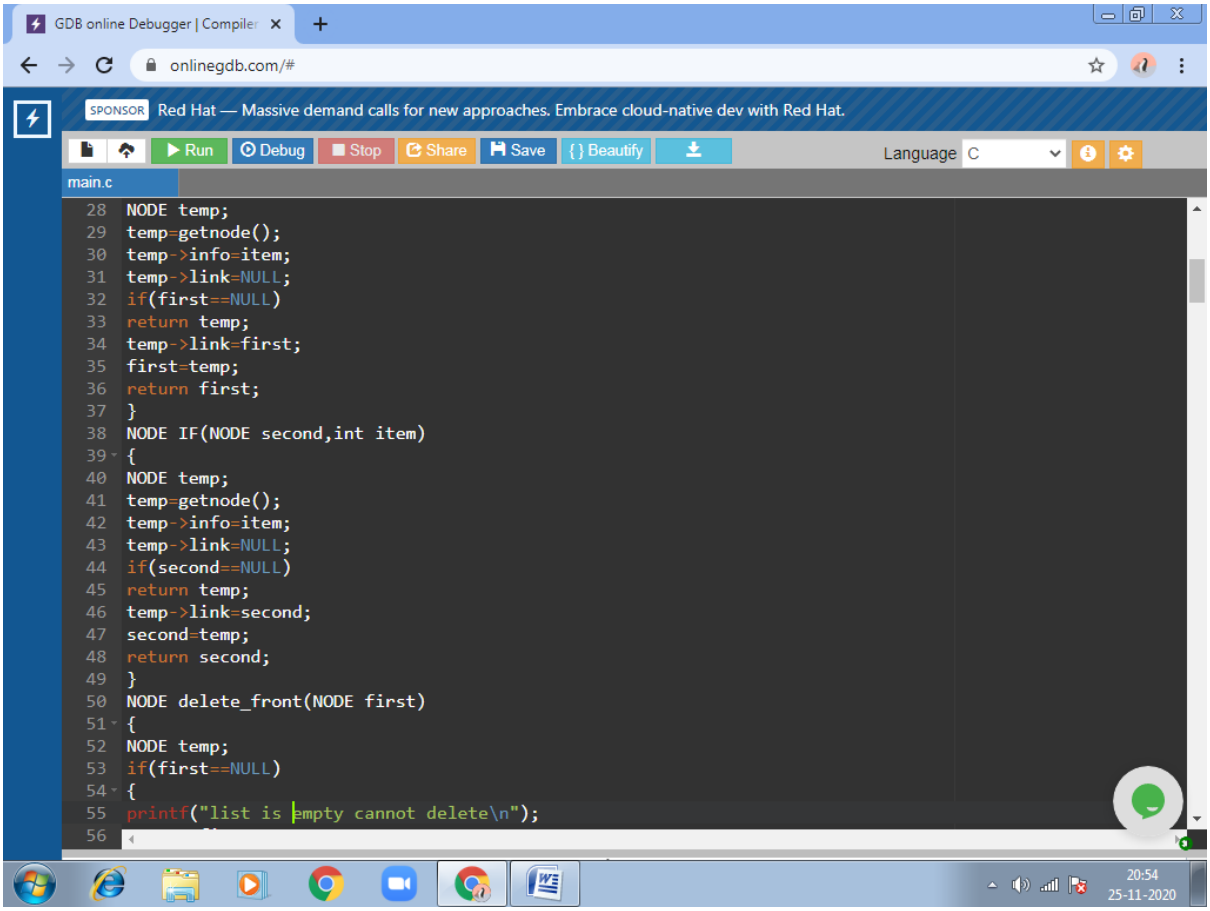
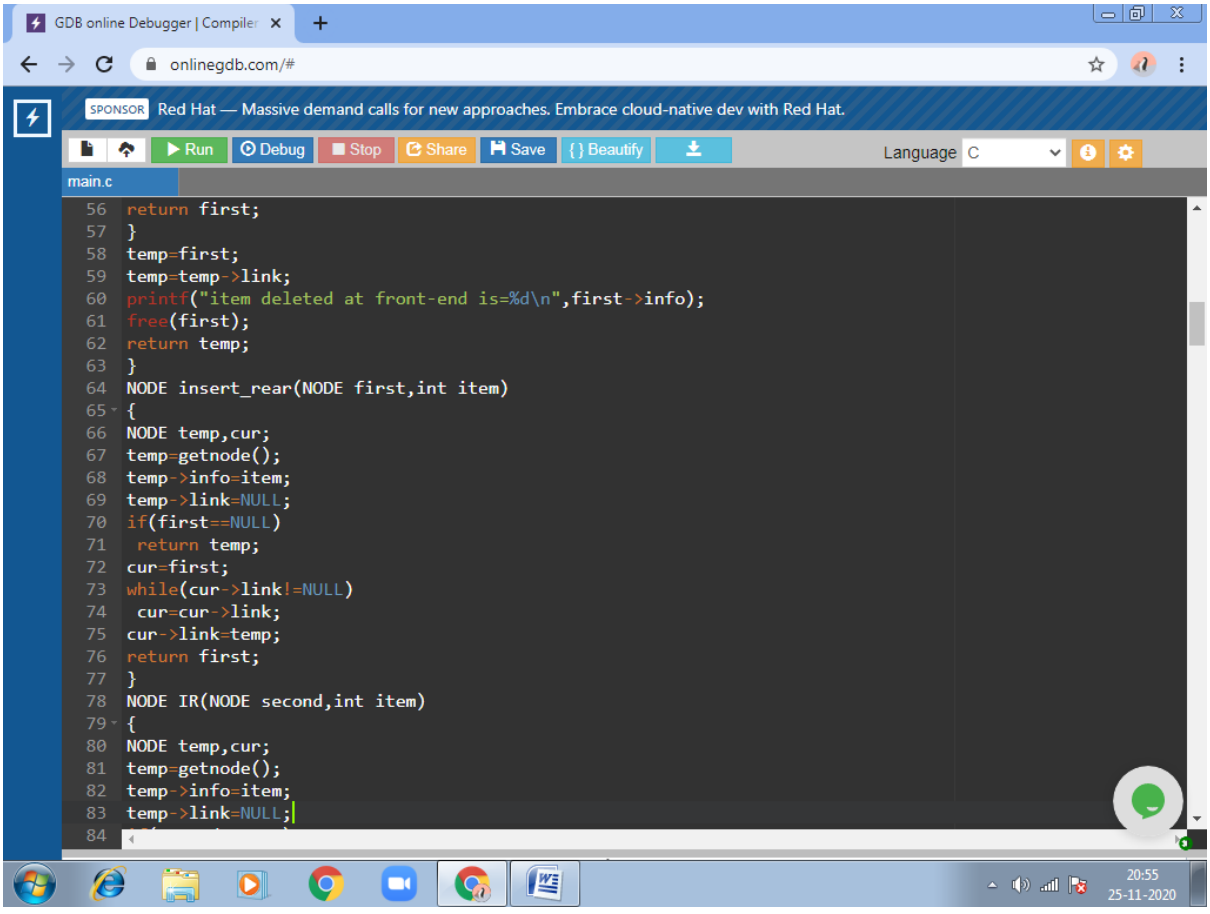
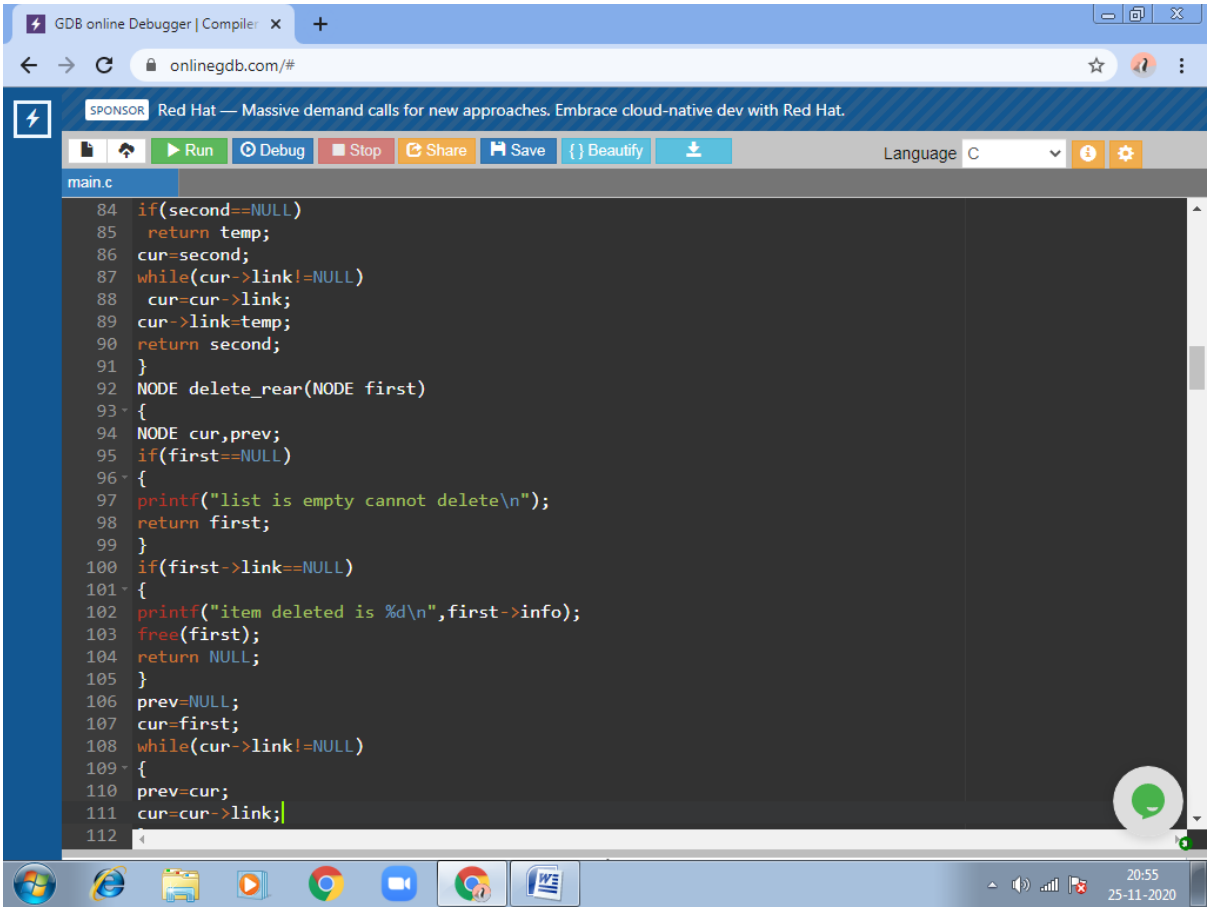
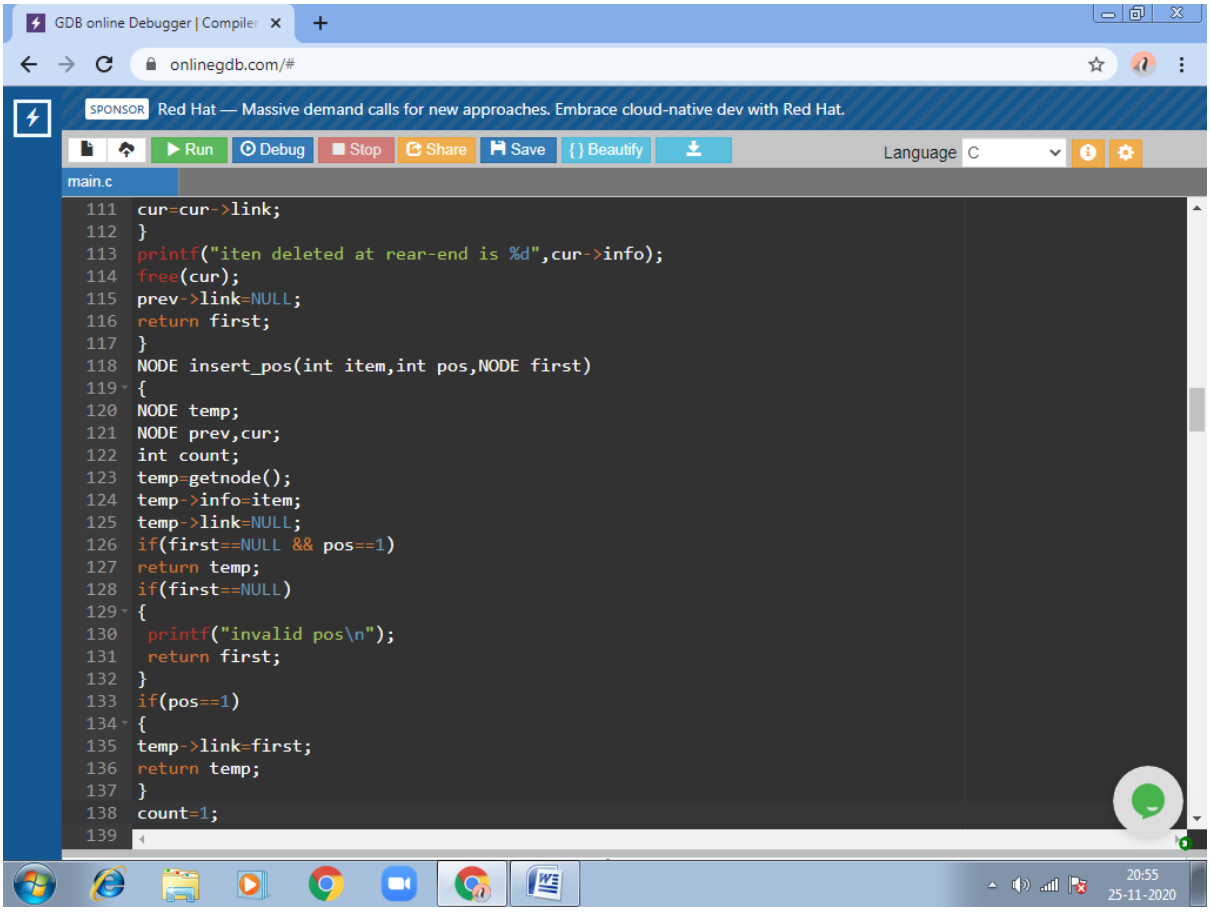
CODE:

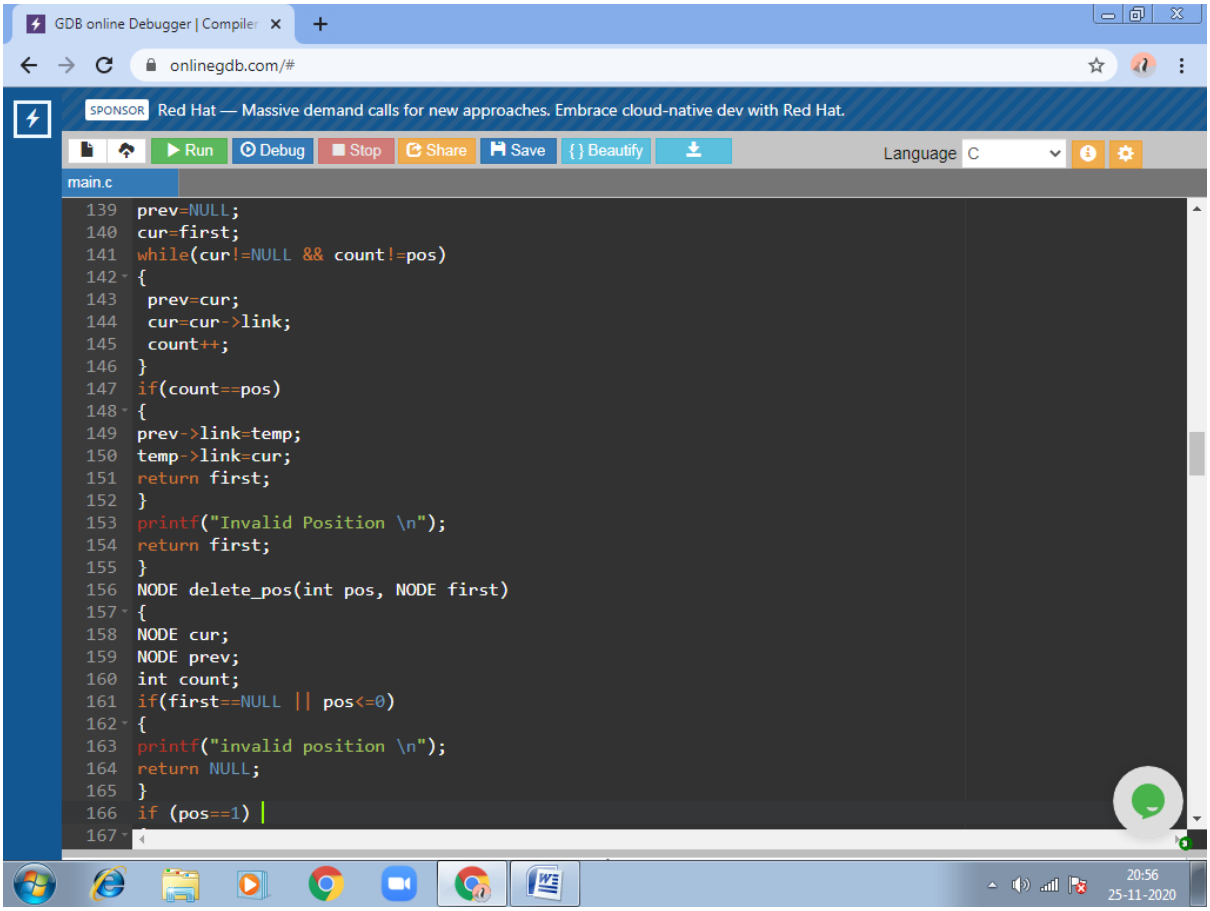

GDB online Debugger | Compiler
onlinegdb.com/#
SPONSOR Red Hat — Massive demand calls for new approaches. Embrace cloud-native dev with Red Hat.
Run
Debug
Stop
Share
Save
{ } Beautify
Language C
main.c
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
struct node
{
int info;
struct node *link;
};
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;
20:54
25-11-2020

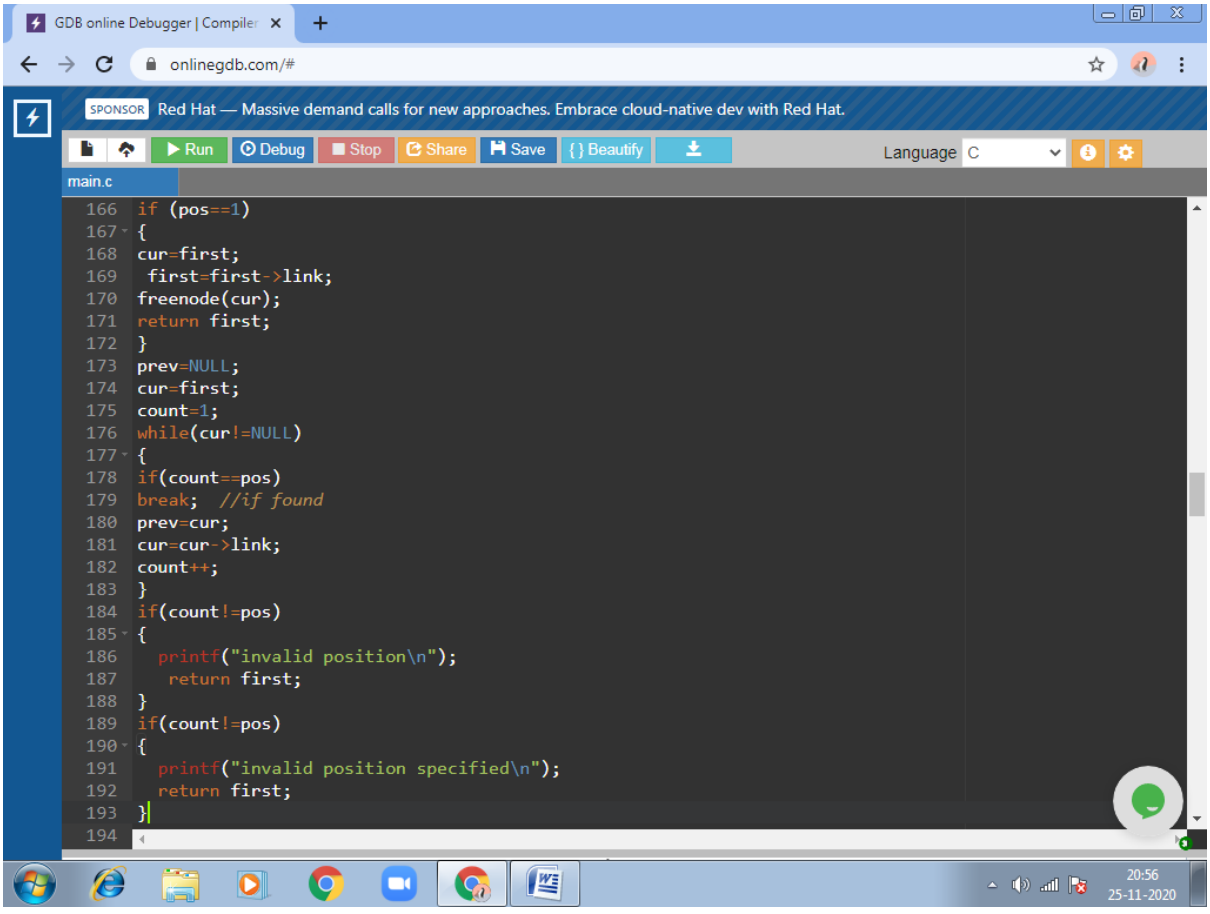

GDB online Debugger | Compiler
onlinegdb.com/#
SPONSOR Red Hat — Massive demand calls for new approaches. Embrace cloud-native dev with Red Hat.
Run
Debug
Stop
Share
Save
{ } Beautify
Language C
main.c
NODE temp;
temp=getnode();
temp->info=item;
temp->link=NULL;
if(first==NULL)
return temp;
temp->link=first;
first=temp;
return first;
}
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 delete_front(NODE first)
{
NODE temp;
if(first==NULL)
{
printf("list is empty cannot delete\n");
20:54
25-11-2020

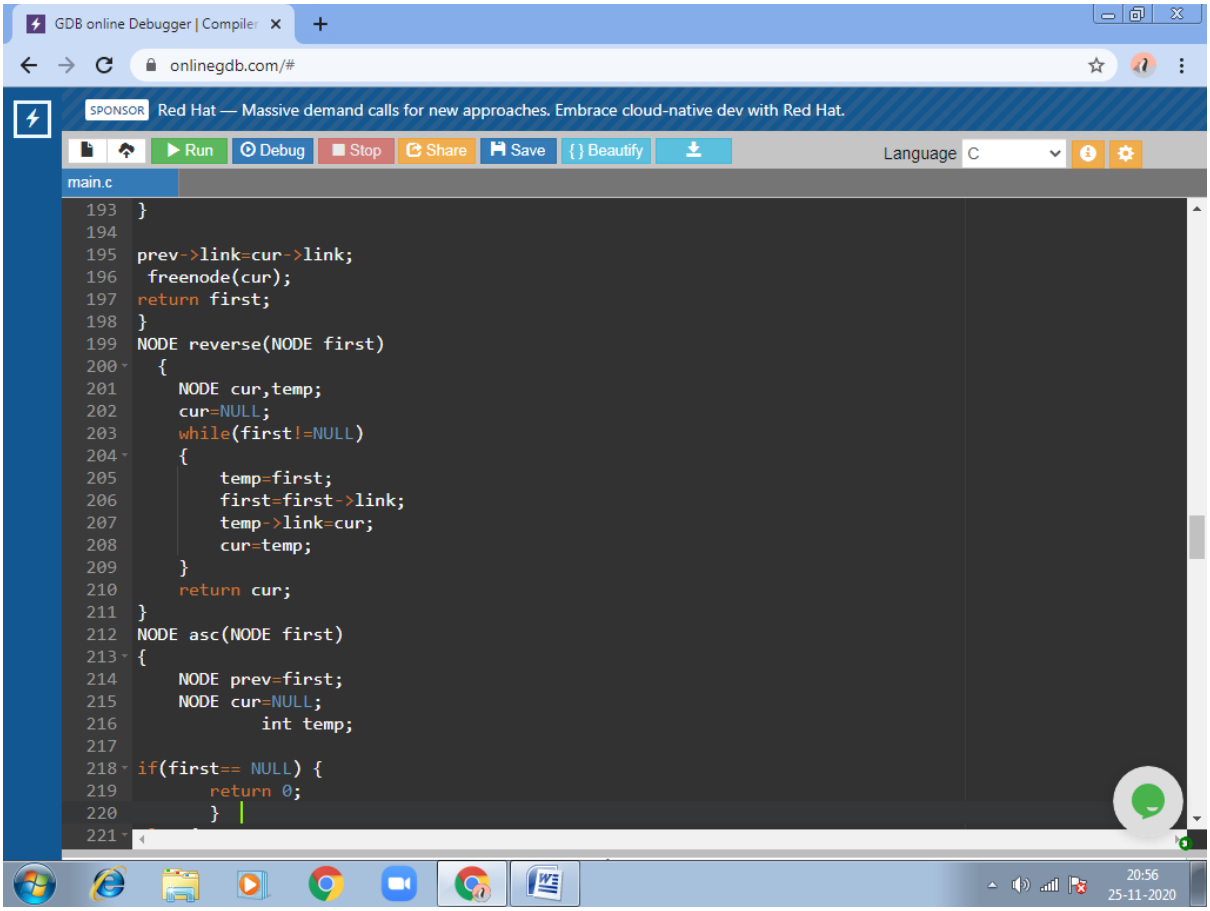

GDB online Debugger | Compiler
onlinegdb.com/#
SPONSOR Red Hat — Massive demand calls for new approaches. Embrace cloud-native dev with Red Hat.
Run
Debug
Stop
Share
Save
{ } Beautify
Language C
main.c
return first;
}
temp=first;
temp=temp->link;
printf("item deleted at front-end is=%d\n",first->info);
free(first);
return temp;
}
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 IR(NODE second,int item)
{
NODE temp,cur;
temp=getnode();
temp->info=item;
temp->link=NULL;
20:55
25-11-2020

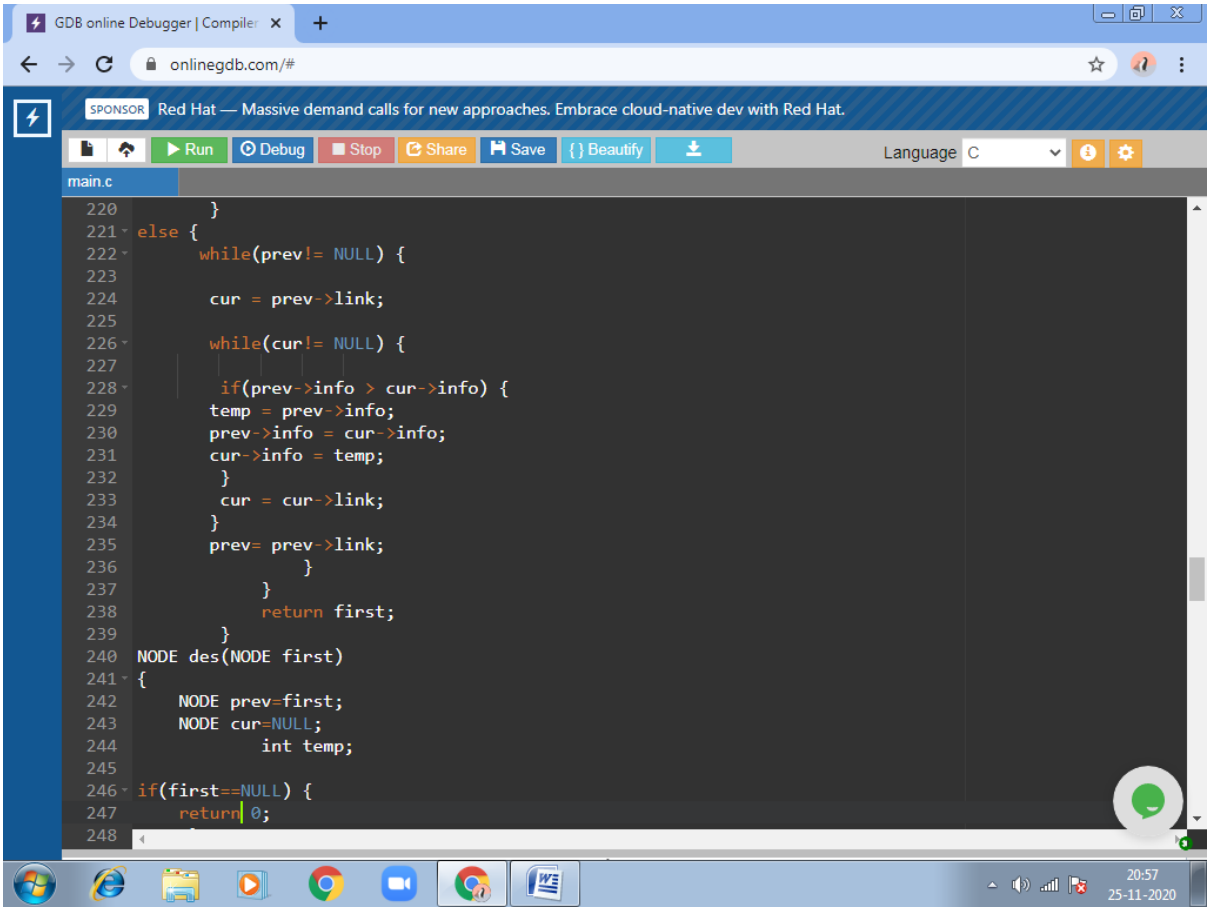

GDB online Debugger | Compiler
onlinegdb.com/#
SPONSOR Red Hat — Massive demand calls for new approaches. Embrace cloud-native dev with Red Hat.
Run Debug Stop Share Save { } Beautify
Language C
main.c
84 if(second==NULL)
85 return temp;
86 cur=second;
87 while(cur->link!=NULL)
88 cur=cur->link;
89 cur->link=temp;
90 return second;
91 }
92 NODE delete_rear(NODE first)
93 {
94 NODE cur,prev;
95 if(first==NULL)
96 {
97 printf("list is empty cannot delete\n");
98 return first;
99 }
100 if(first->link==NULL)
101 {
102 printf("item deleted is %d\n",first->info);
103 free(first);
104 return NULL;
105 }
106 prev=NULL;
107 cur=first;
108 while(cur->link!=NULL)
109 {
110 prev=cur;
111 cur=cur->link;
112
20:55
25-11-2020

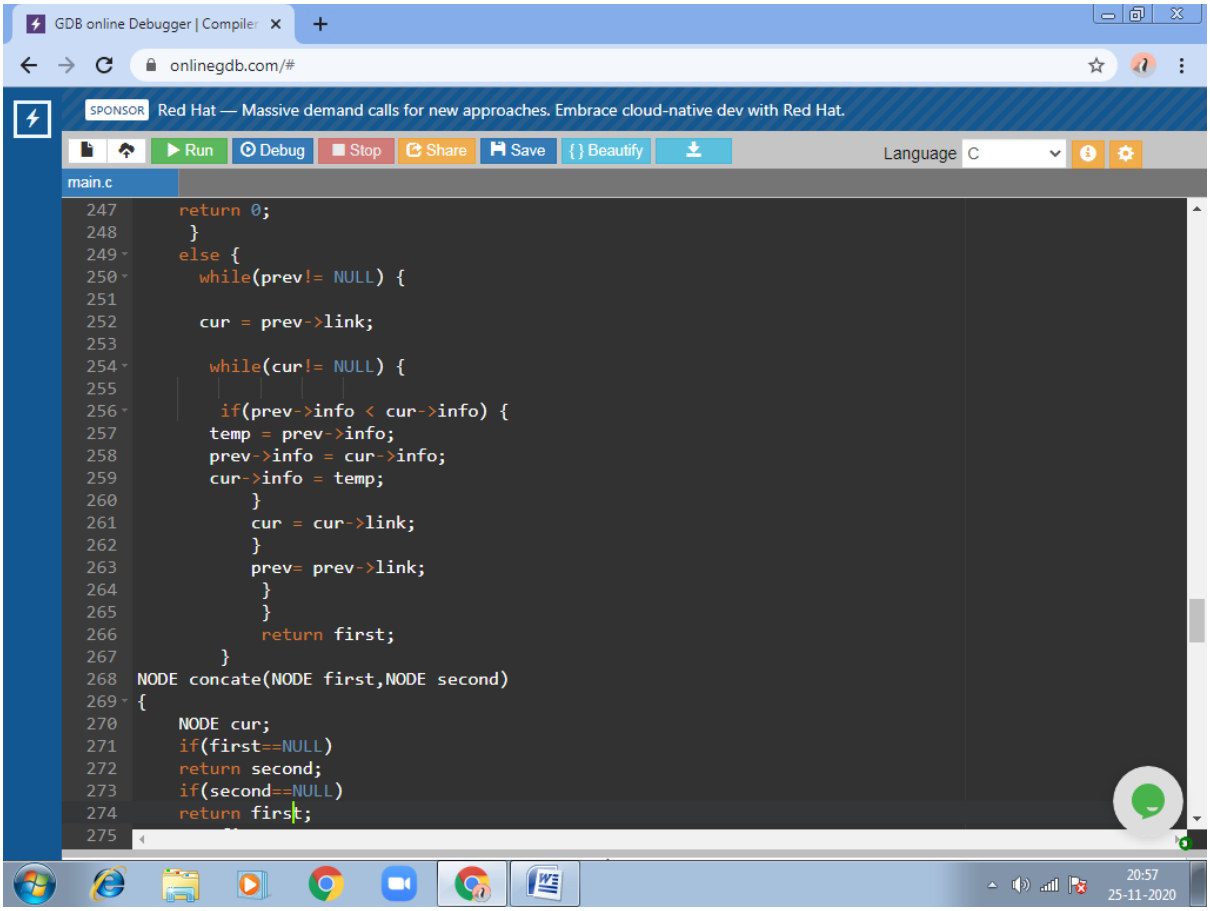

GDB online Debugger | Compiler
onlinegdb.com/#
SPONSOR Red Hat — Massive demand calls for new approaches. Embrace cloud-native dev with Red Hat.
Run Debug Stop Share Save { } Beautify
Language C
main.c
111 cur=cur->link;
112 }
113 printf("iten deleted at rear-end is %d",cur->info);
114 free(cur);
115 prev->link=NULL;
116 return first;
117 }
118 NODE insert_pos(int item,int pos,NODE first)
119 {
120 NODE temp;
121 NODE prev,cur;
122 int count;
123 temp=getnode();
124 temp->info=item;
125 temp->link=NULL;
126 if(first==NULL && pos==1)
127 return temp;
128 if(first==NULL)
129 {
130 printf("invalid pos\n");
131 return first;
132 }
133 if(pos==1)
134 {
135 temp->link=first;
136 return temp;
137 }
138 count=1;
139
20:55
25-11-2020

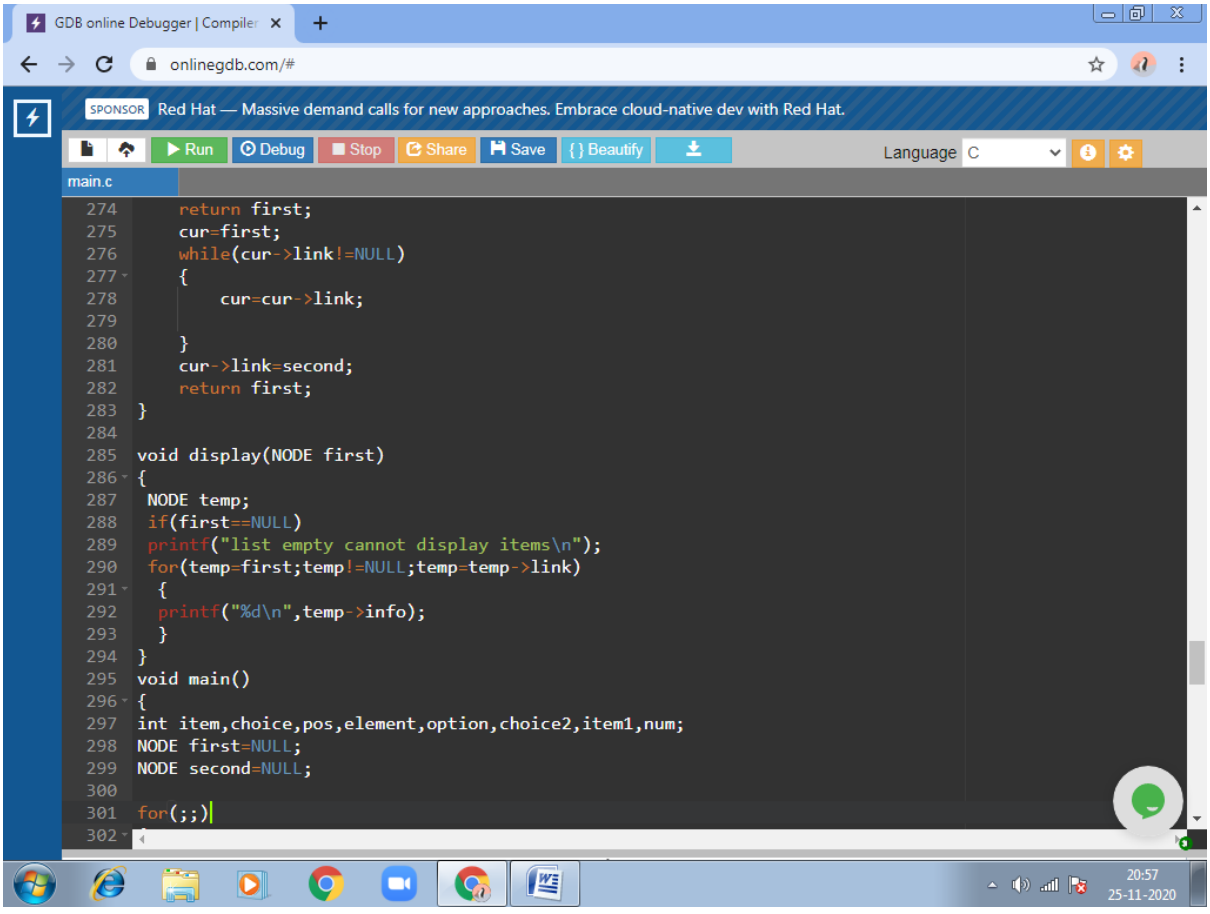

GDB online Debugger | Compiler
onlinegdb.com/#
SPONSOR Red Hat — Massive demand calls for new approaches. Embrace cloud-native dev with Red Hat.
Run Debug Stop Share Save { } Beautify
Language C
main.c
prev=NULL;
cur=first;
while(cur!=NULL && count!=pos)
{
prev=cur;
cur=cur->link;
count++;
}
if(count==pos)
{
prev->link=temp;
temp->link=cur;
return first;
}
printf("Invalid Position \n");
return first;
}
NODE delete_pos(int pos, NODE first)
{
NODE cur;
NODE prev;
int count;
if(first==NULL || pos<=0)
{
printf("invalid position \n");
return NULL;
}
if (pos==1)
20:56
25-11-2020

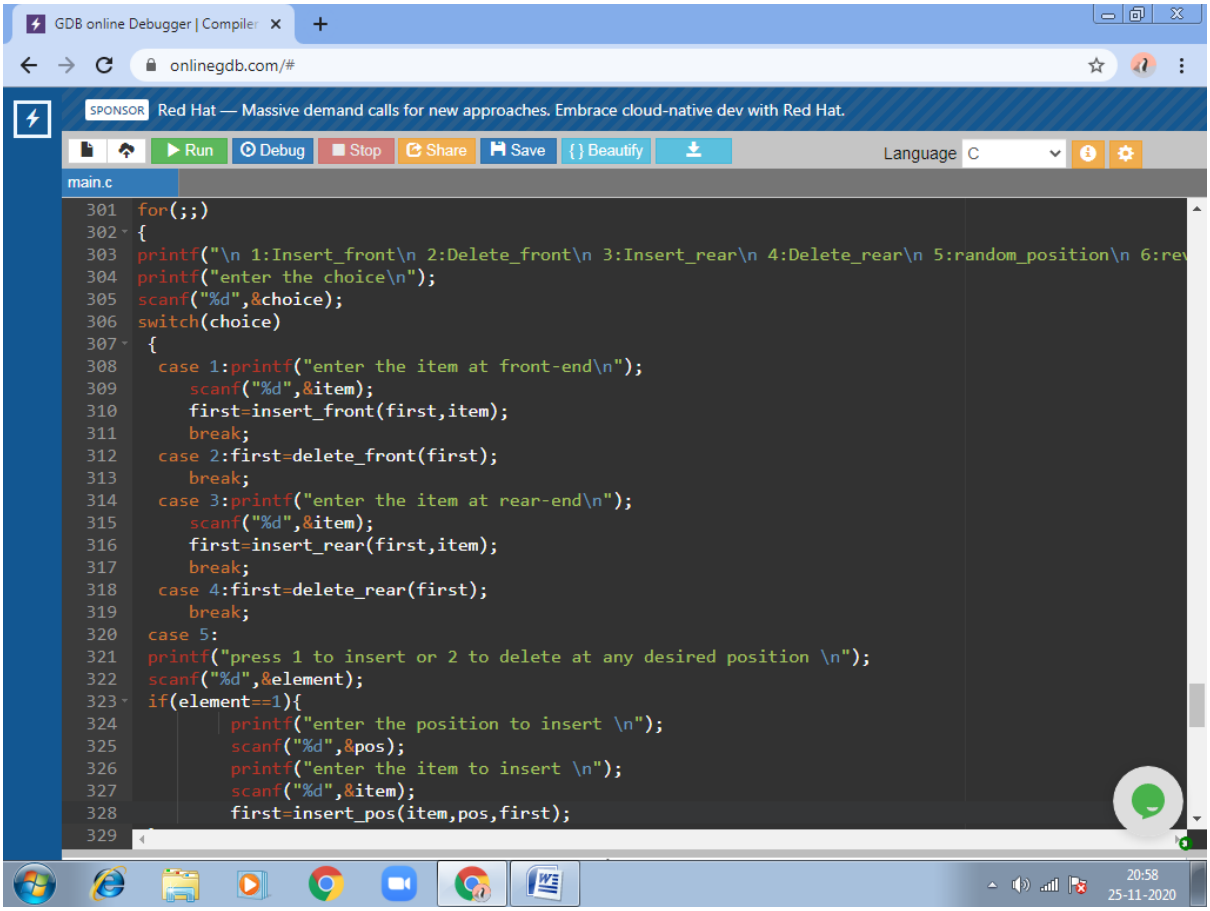

GDB online Debugger | Compiler
onlinegdb.com/#
SPONSOR Red Hat — Massive demand calls for new approaches. Embrace cloud-native dev with Red Hat.
Run Debug Stop Share Save { } Beautify
Language C
main.c
if (pos==1)
{
cur=first;
first=first->link;
freenode(cur);
return first;
}
prev=NULL;
cur=first;
count=1;
while(cur!=NULL)
{
if(count==pos)
break; //if found
prev=cur;
cur=cur->link;
count++;
}
if(count!=pos)
{
printf("invalid position\n");
return first;
}
if(count!=pos)
{
printf("invalid position specified\n");
return first;
}
20:56
25-11-2020

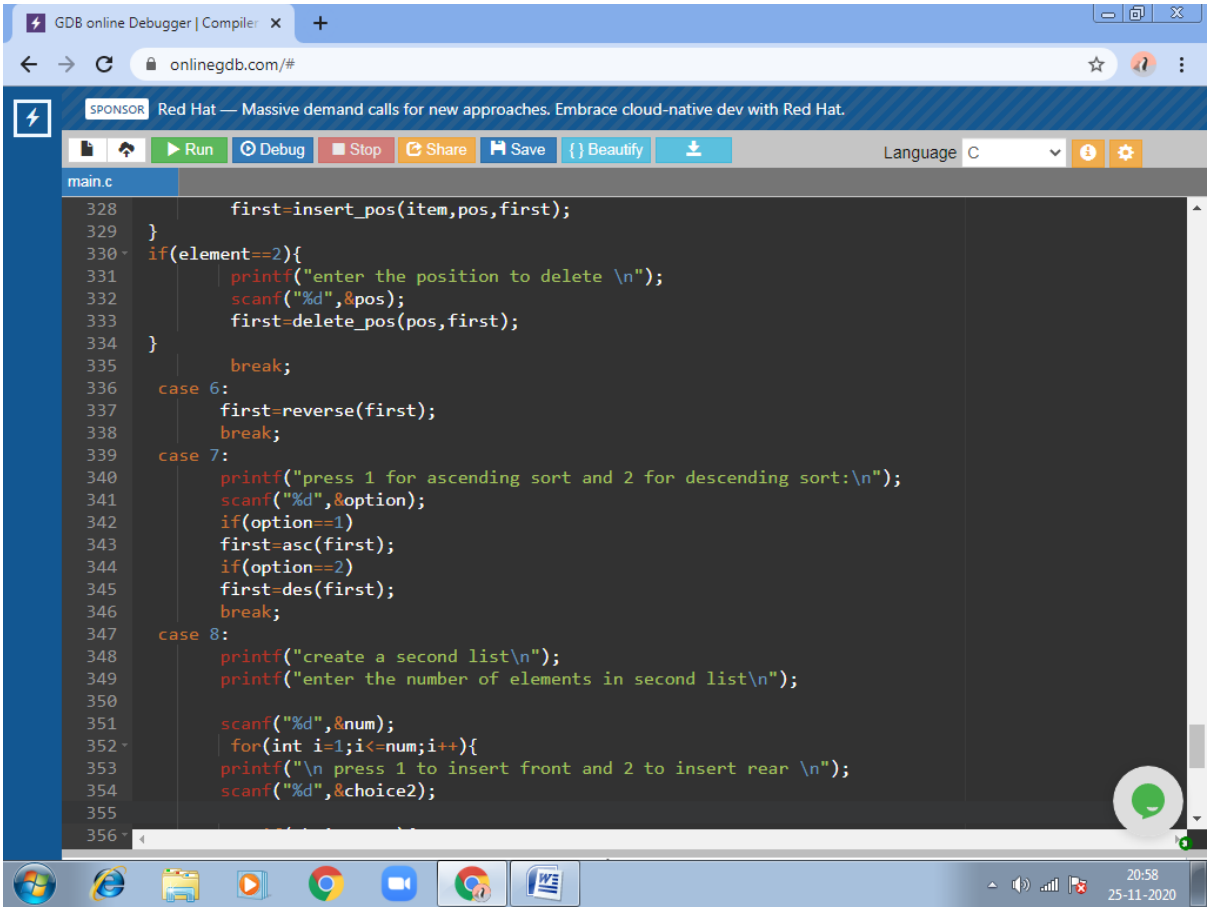

GDB online Debugger | Compiler
onlinegdb.com/#
SPONSOR Red Hat — Massive demand calls for new approaches. Embrace cloud-native dev with Red Hat.
Run Debug Stop Share Save { } Beautify
Language C
main.c
}
prev->link=cur->link;
freenode(cur);
return first;
}
NODE reverse(NODE first)
{
NODE cur,temp;
cur=NULL;
while(first!=NULL)
{
temp=first;
first=first->link;
temp->link=cur;
cur=temp;
}
return cur;
}
NODE asc(NODE first)
{
NODE prev=first;
NODE cur=NULL;
int temp;
if(first== NULL) {
return 0;
}
20:56
25-11-2020

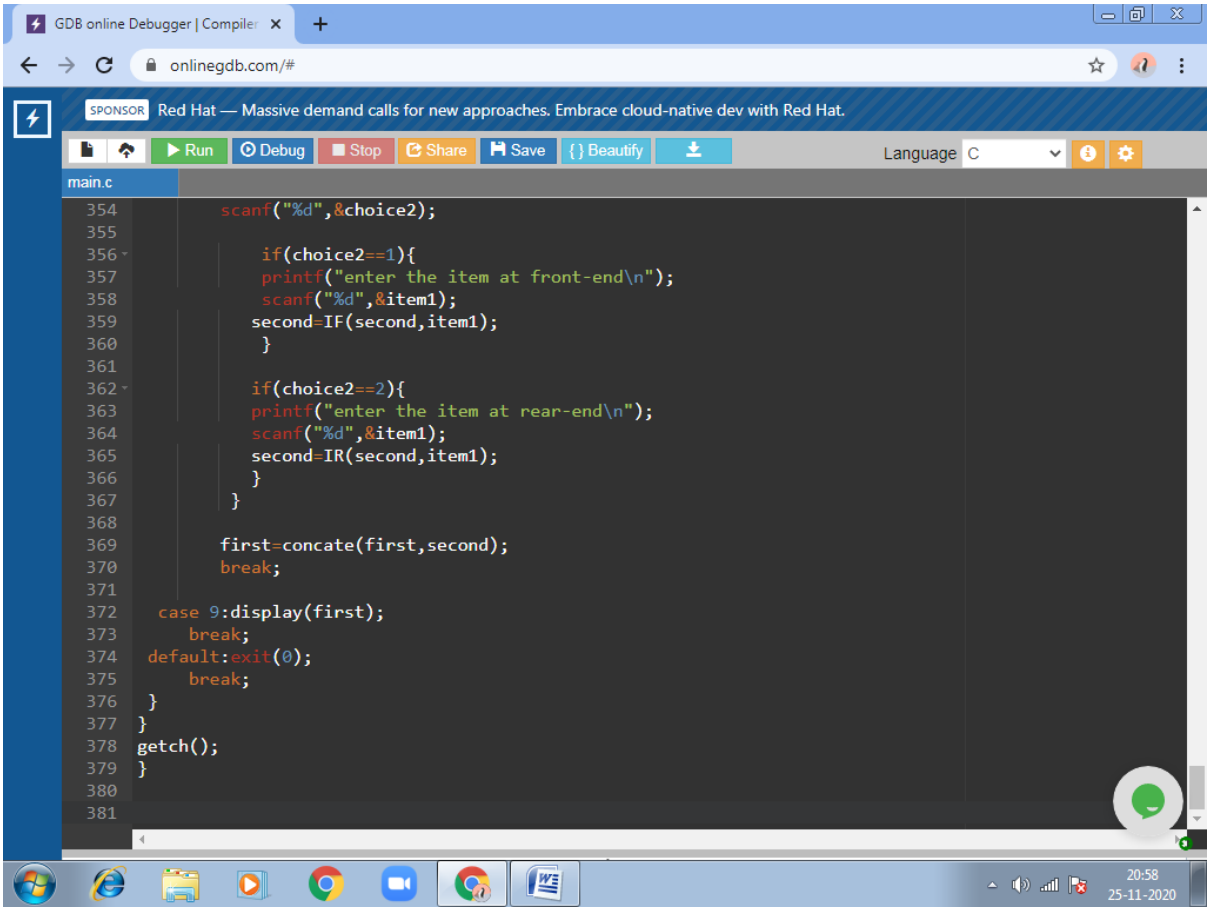

GDB online Debugger | Compiler
onlinegdb.com/#
SPONSOR Red Hat — Massive demand calls for new approaches. Embrace cloud-native dev with Red Hat.
Run Debug Stop Share Save { } Beautify
Language C
main.c
}
else {
while(prev!= NULL) {
cur = prev->link;
while(cur!= NULL) {
if(prev->info > cur->info) {
temp = prev->info;
prev->info = cur->info;
cur->info = temp;
}
cur = cur->link;
}
prev= prev->link;
}
}
return first;
}
NODE des(NODE first)
{
NODE prev=first;
NODE cur=NULL;
int temp;
if(first==NULL) {
return 0;
20:57
25-11-2020


GDB online Debugger | Compiler
onlinegdb.com/#
SPONSOR Red Hat — Massive demand calls for new approaches. Embrace cloud-native dev with Red Hat.
Run Debug Stop Share Save { } Beautify
Language C
main.c
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 concate(NODE first,NODE second)
{
NODE cur;
if(first==NULL)
return second;
if(second==NULL)
return first;
20:57
25-11-2020

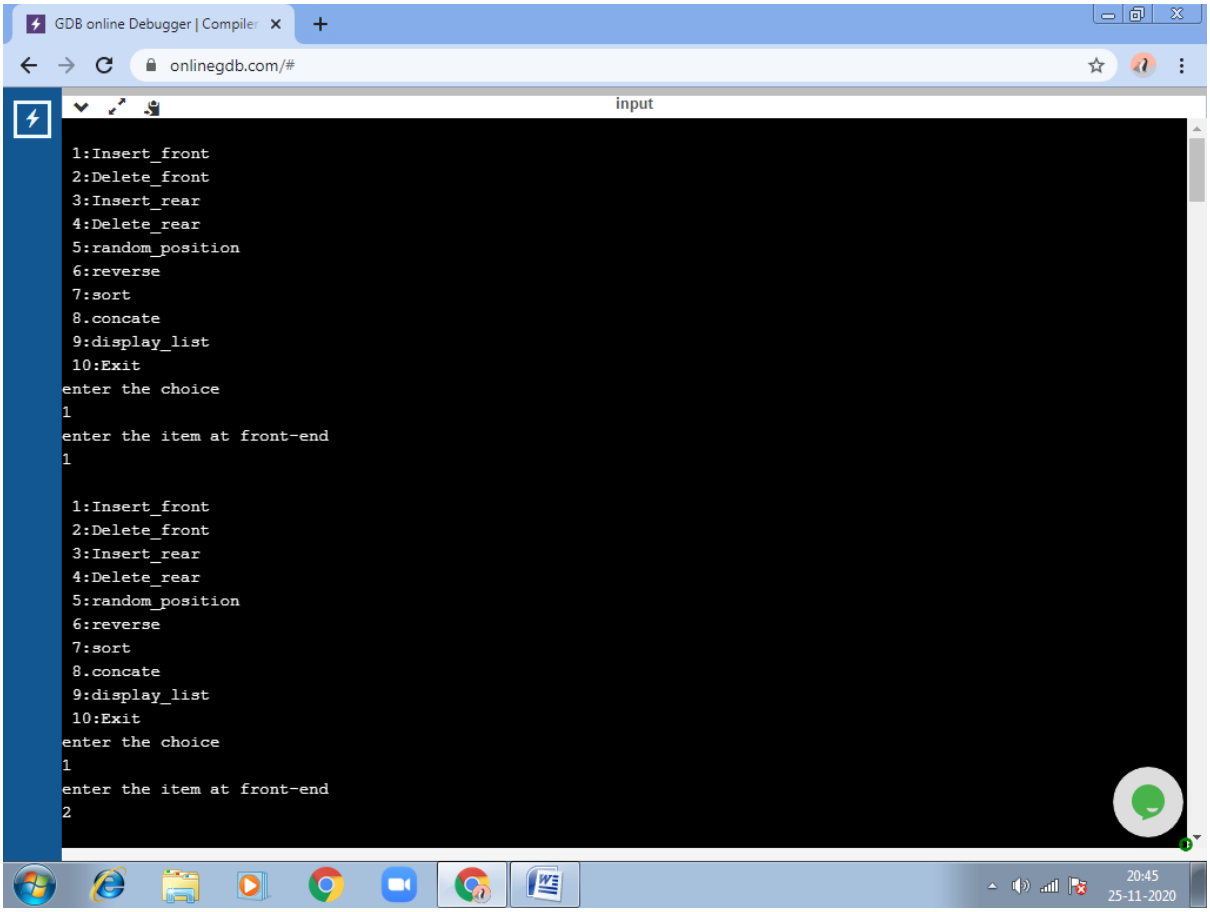

GDB online Debugger | Compiler
onlinegdb.com/#
SPONSOR Red Hat — Massive demand calls for new approaches. Embrace cloud-native dev with Red Hat.
Run Debug Stop Share Save { } Beautify
Language C
main.c
return first;
cur=first;
while(cur->link!=NULL)
{
cur=cur->link;
}
cur->link=second;
return first;
}
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);
}
}
void main()
{
int item,choice,pos,element,option,choice2,item1,num;
NODE first=NULL;
NODE second=NULL;
for(;;)
20:57
25-11-2020

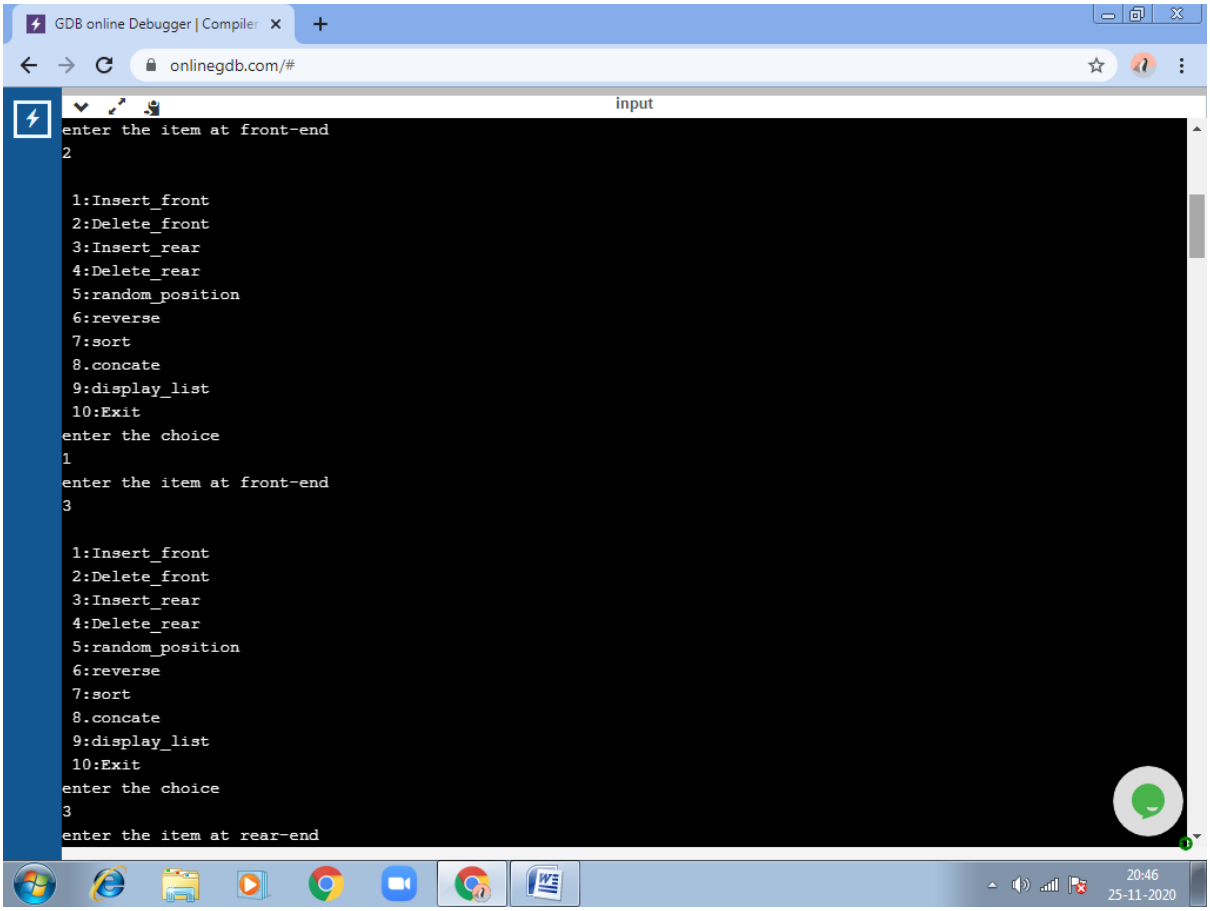

GDB online Debugger | Compiler
onlinegdb.com/#
SPONSOR Red Hat — Massive demand calls for new approaches. Embrace cloud-native dev with Red Hat.
Run Debug Stop Share Save { } Beautify
Language C
main.c
for(;;)
{
printf("\n 1:Insert_front\n 2:Delete_front\n 3:Insert_rear\n 4:Delete_rear\n 5:random_position\n 6:rev
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:printf("enter the item at rear-end\n");
scanf("%d",&item);
first=insert_rear(first,item);
break;
case 4:first=delete_rear(first);
break;
case 5:
printf("press 1 to insert or 2 to delete at any desired position \n");
scanf("%d",&element);
if(element==1){
printf("enter the position to insert \n");
scanf("%d",&pos);
printf("enter the item to insert \n");
scanf("%d",&item);
first=insert_pos(item,pos,first);
20:58
25-11-2020

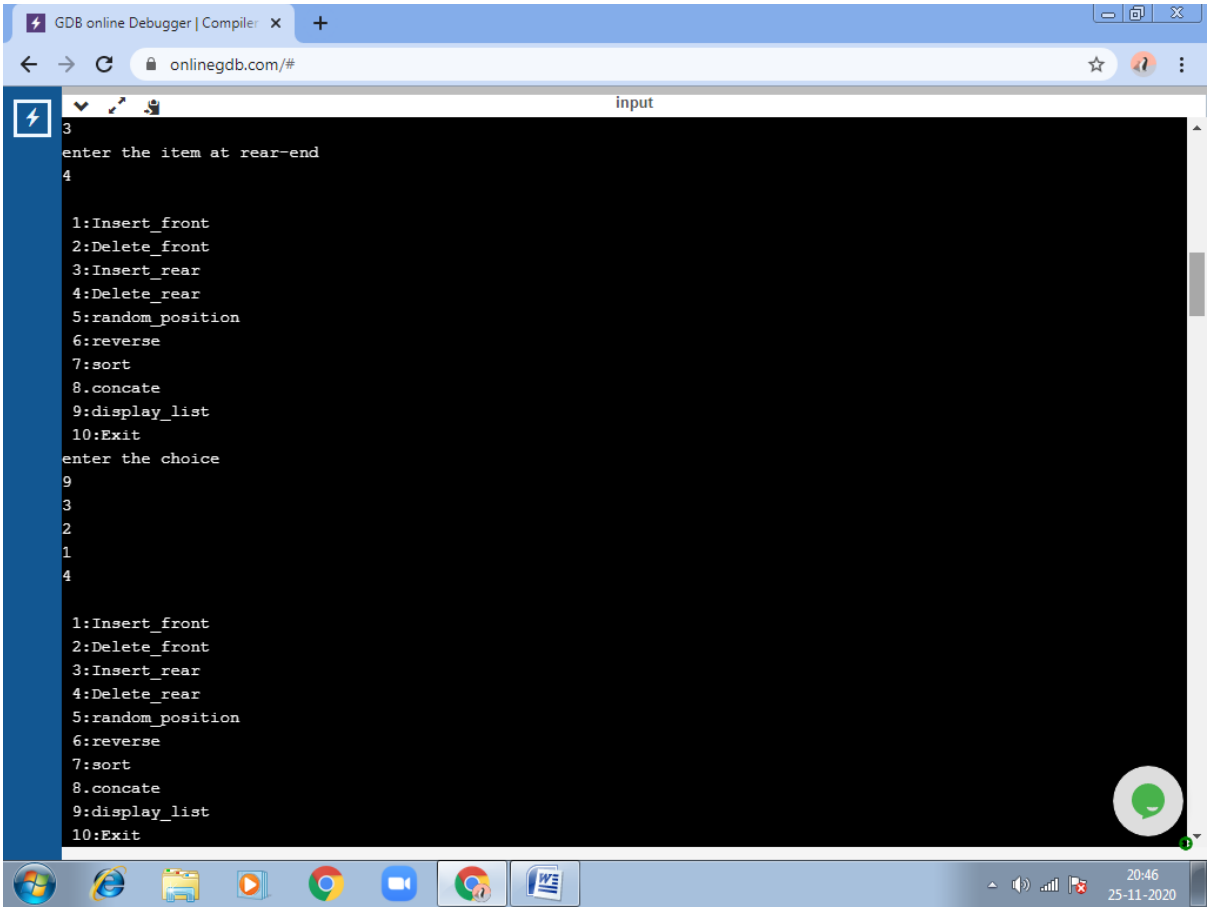

GDB online Debugger | Compiler
onlinegdb.com/#
SPONSOR Red Hat — Massive demand calls for new approaches. Embrace cloud-native dev with Red Hat.
Run Debug Stop Share Save { } Beautify
Language C
main.c
first=insert_pos(item,pos,first);
}
if(element==2){
printf("enter the position to delete \n");
scanf("%d",&pos);
first=delete_pos(pos,first);
}
break;
case 6:
first=reverse(first);
break;
case 7:
printf("press 1 for ascending sort and 2 for descending sort:\n");
scanf("%d",&option);
if(option==1)
first=asc(first);
if(option==2)
first=des(first);
break;
case 8:
printf("create a second list\n");
printf("enter the number of elements in second list\n");
scanf("%d",&num);
for(int i=1;i<=num;i++){
printf("\n press 1 to insert front and 2 to insert rear \n");
scanf("%d",&choice2);
20:58
25-11-2020

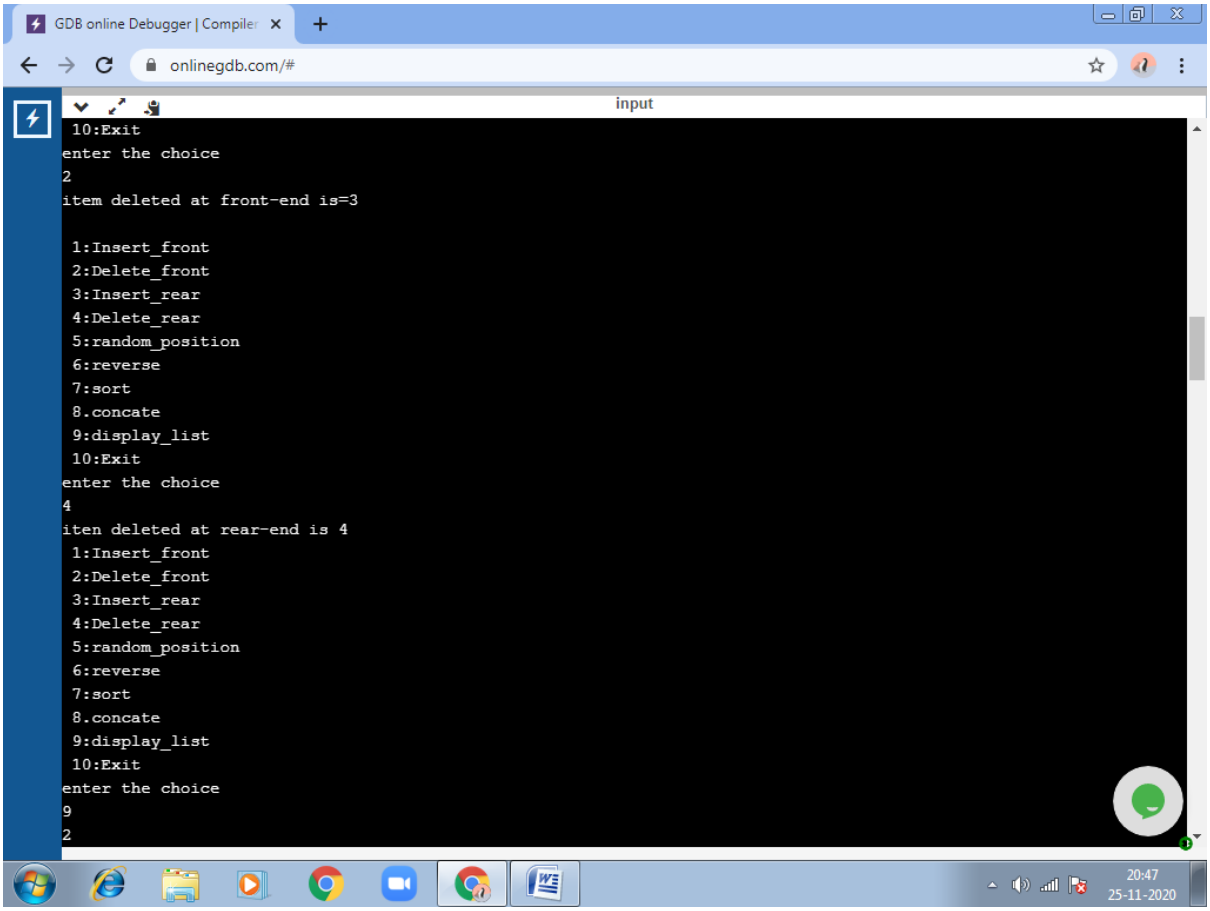

GDB online Debugger | Compiler
onlinegdb.com/#
SPONSOR Red Hat — Massive demand calls for new approaches. Embrace cloud-native dev with Red Hat.
Run
Debug
Stop
Share
Save
{ } Beautify
Language C
main.c
scanf("%d",&choice2);
if(choice2==1){
printf("enter the item at front-end\n");
scanf("%d",&item1);
second=IF(second,item1);
}
if(choice2==2){
printf("enter the item at rear-end\n");
scanf("%d",&item1);
second=IR(second,item1);
}
}
first=concate(first,second);
break;
case 9:display(first);
break;
default:exit(0);
break;
}
}
getch();
}
20:58
25-11-2020

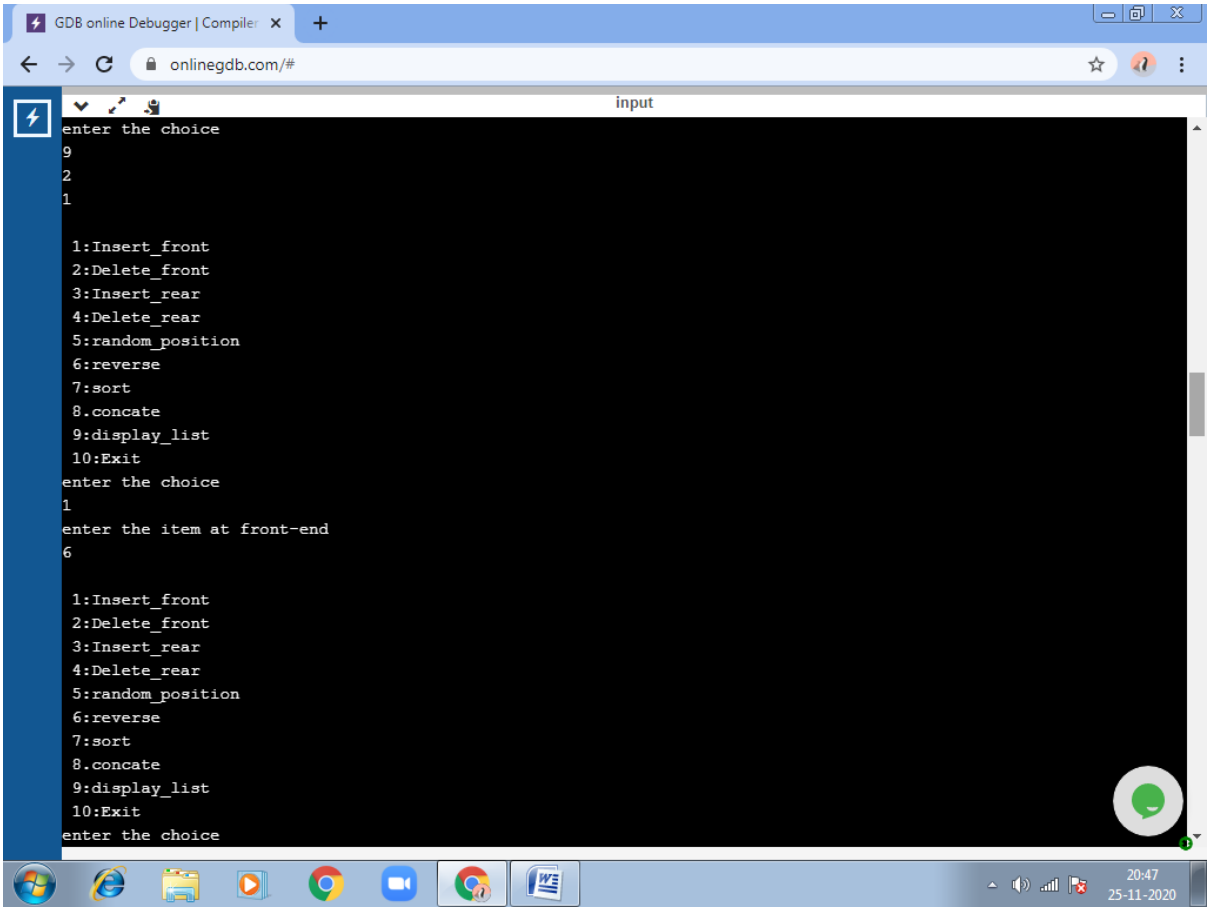
OUTPUT:

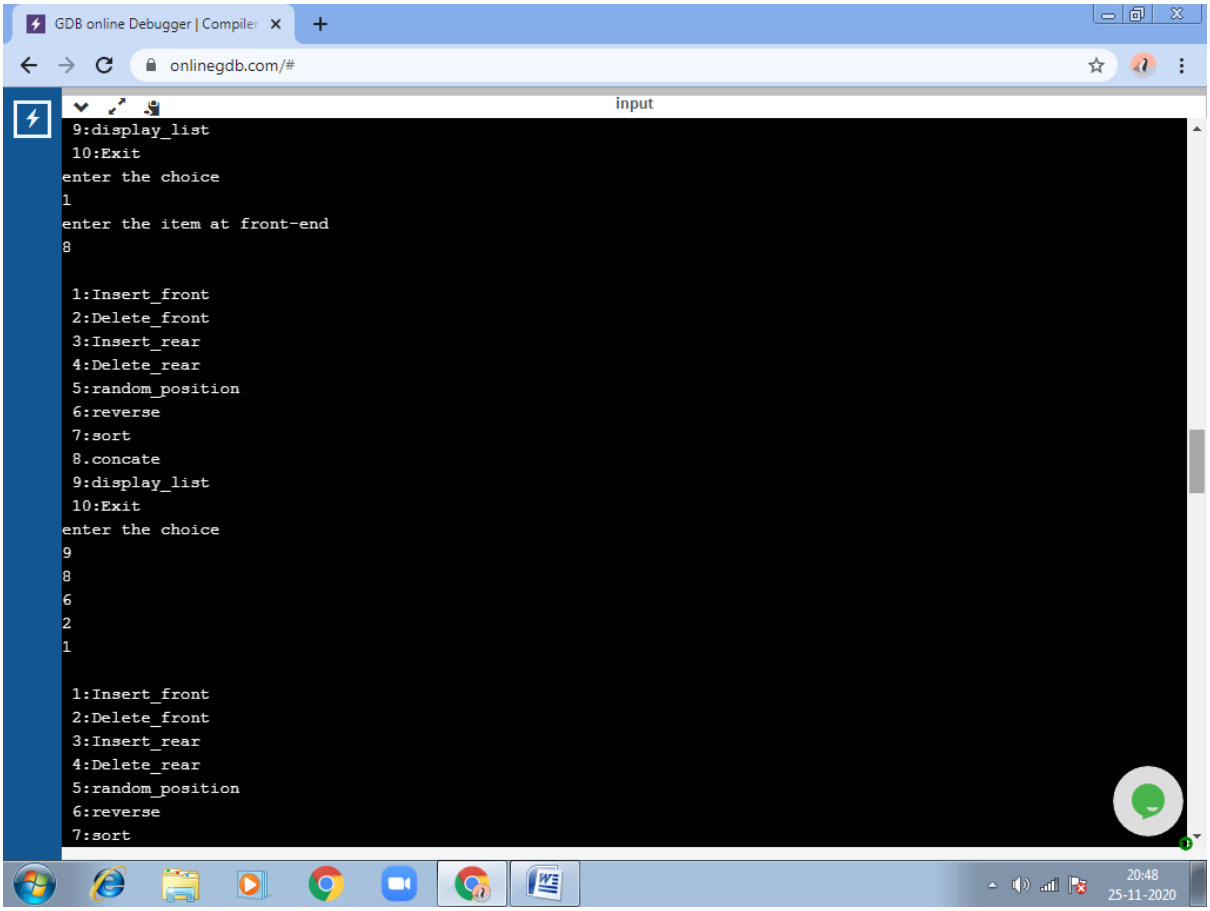

GDB online Debugger | Compiler
onlinegdb.com/#
input
1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:random_position
6:reverse
7:sort
8.concate
9:display_list
10:Exit
enter the choice
1
enter the item at front-end
1
1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:random_position
6:reverse
7:sort
8.concate
9:display_list
10:Exit
enter the choice
1
enter the item at front-end
2
20:45
25-11-2020

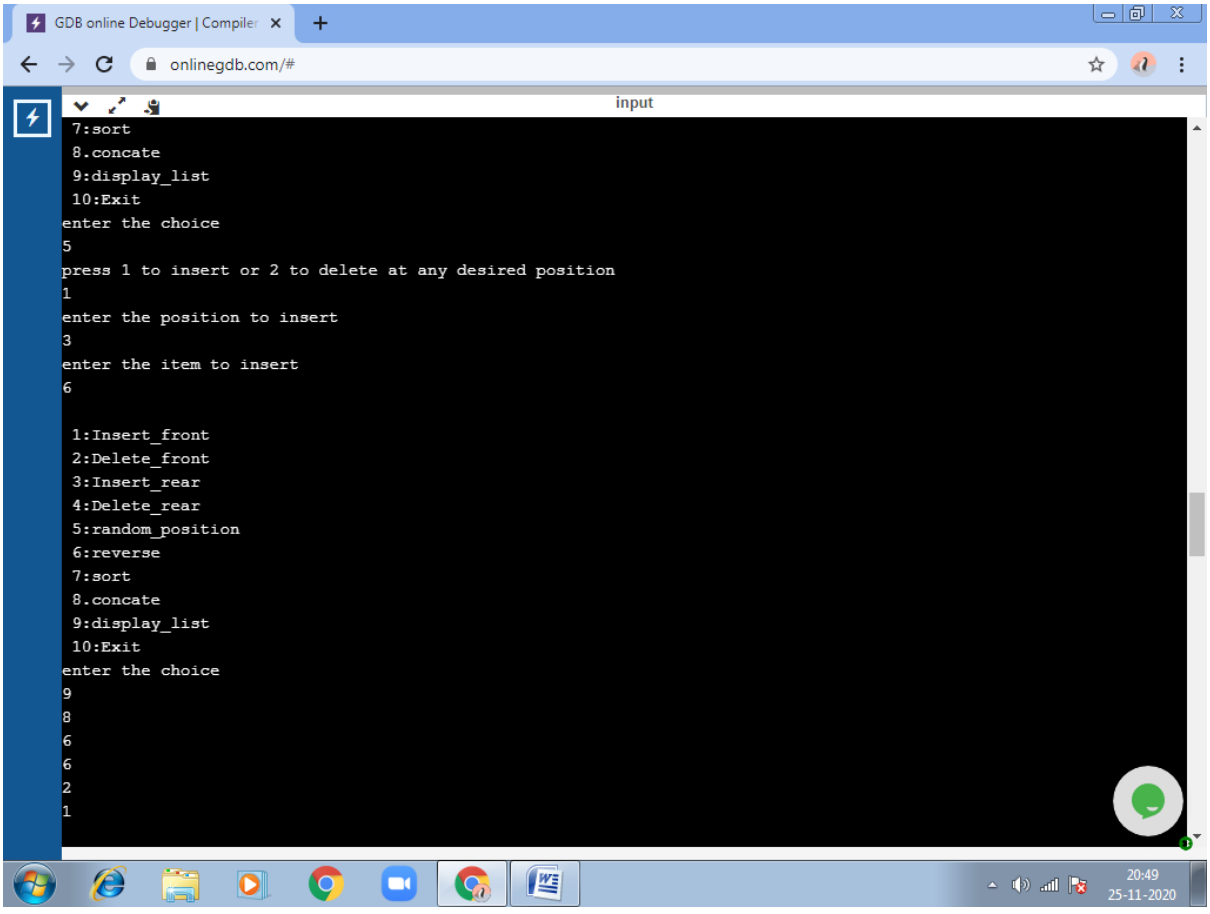

GDB online Debugger | Compiler
onlinegdb.com/#
input
enter the item at front-end
2
1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:random_position
6:reverse
7:sort
8.concate
9:display_list
10:Exit
enter the choice
1
enter the item at front-end
3
1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:random_position
6:reverse
7:sort
8.concate
9:display_list
10:Exit
enter the choice
3
enter the item at rear-end
20:46
25-11-2020

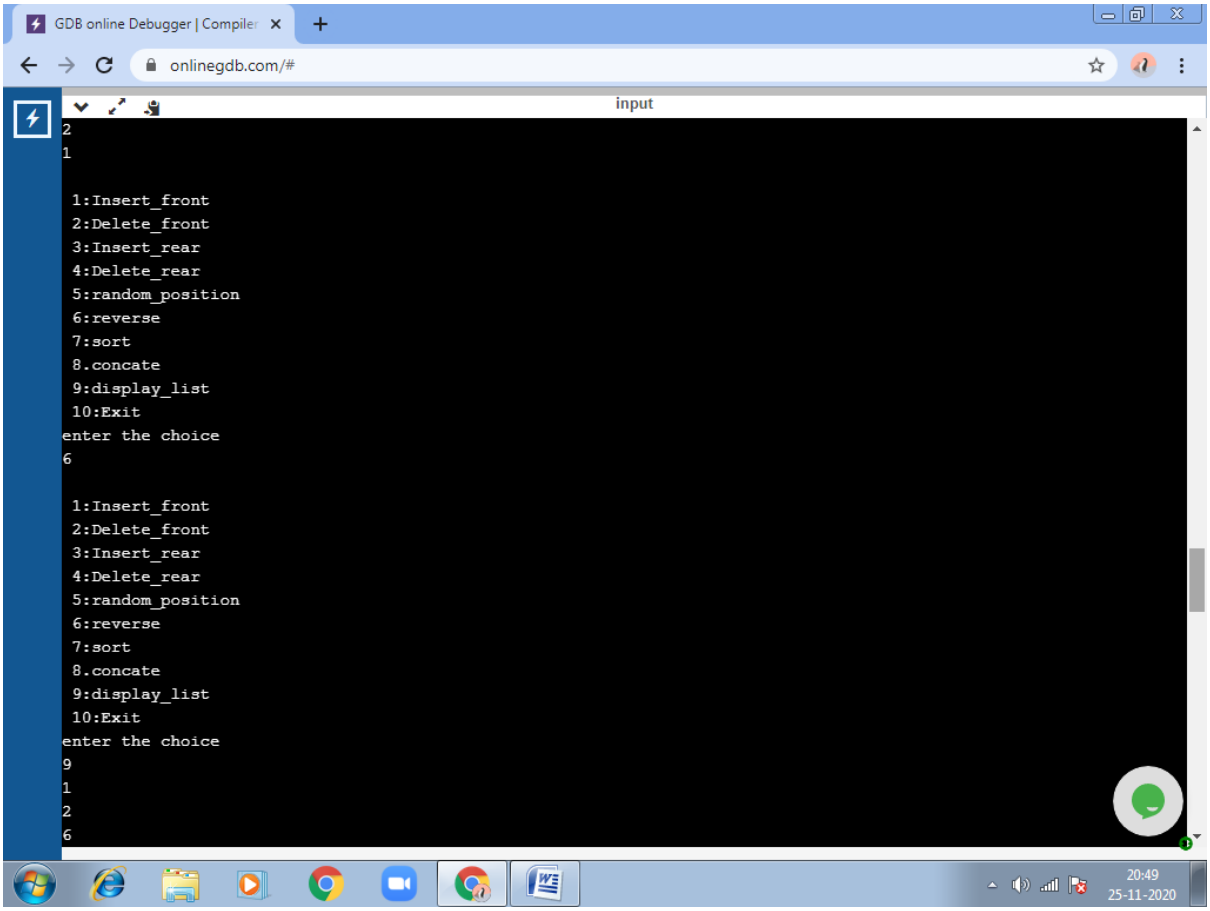

GDB online Debugger | Compiler
onlinegdb.com/#
input
3
enter the item at rear-end
4
1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:random_position
6:reverse
7:sort
8.concate
9:display_list
10:Exit
enter the choice
9
3
2
1
4
1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:random_position
6:reverse
7:sort
8.concate
9:display_list
10:Exit
20:46
25-11-2020

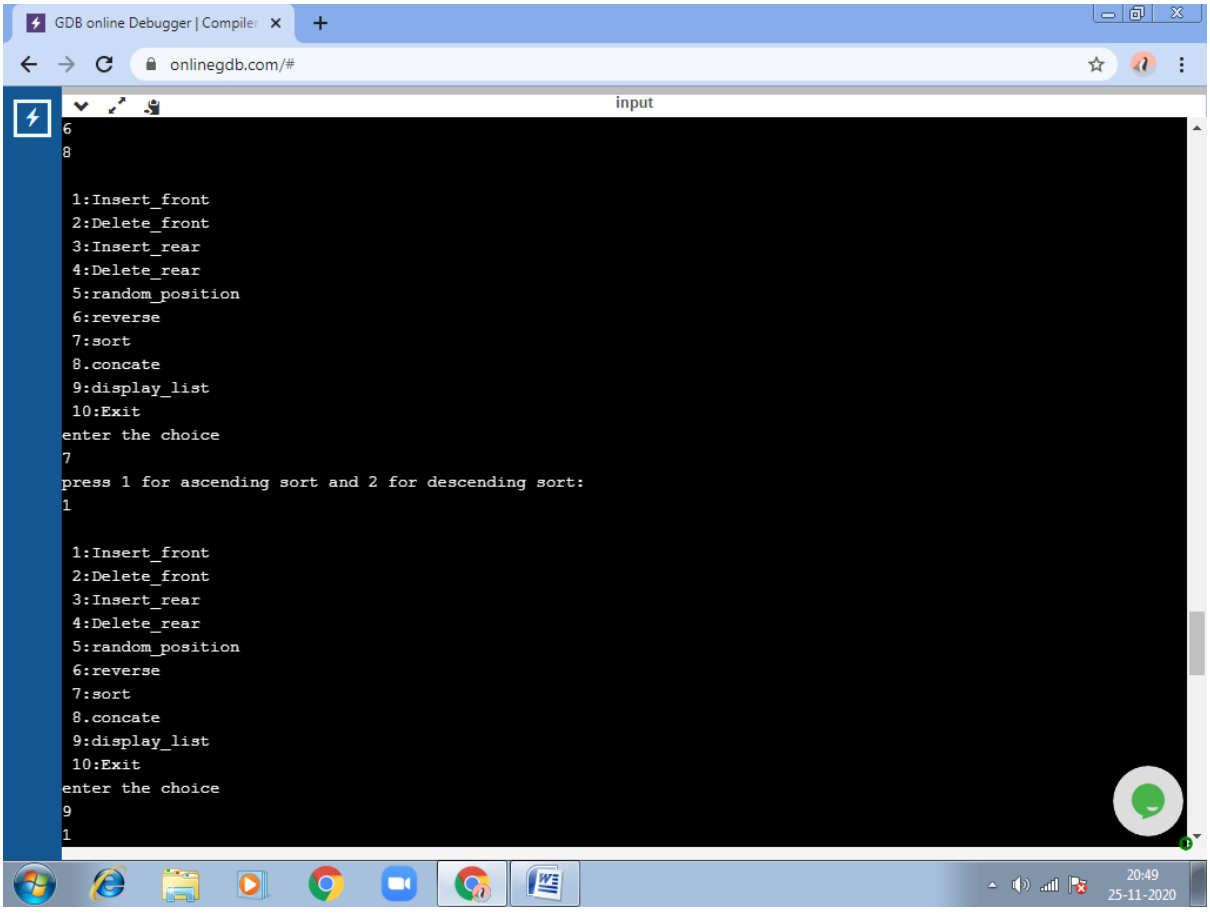

GDB online Debugger | Compiler
onlinegdb.com/#
input
10:Exit
enter the choice
2
item deleted at front-end is=3
1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:random_position
6:reverse
7:sort
8.concate
9:display_list
10:Exit
enter the choice
4
iten deleted at rear-end is 4
1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:random_position
6:reverse
7:sort
8.concate
9:display_list
10:Exit
enter the choice
9
2
20:47
25-11-2020

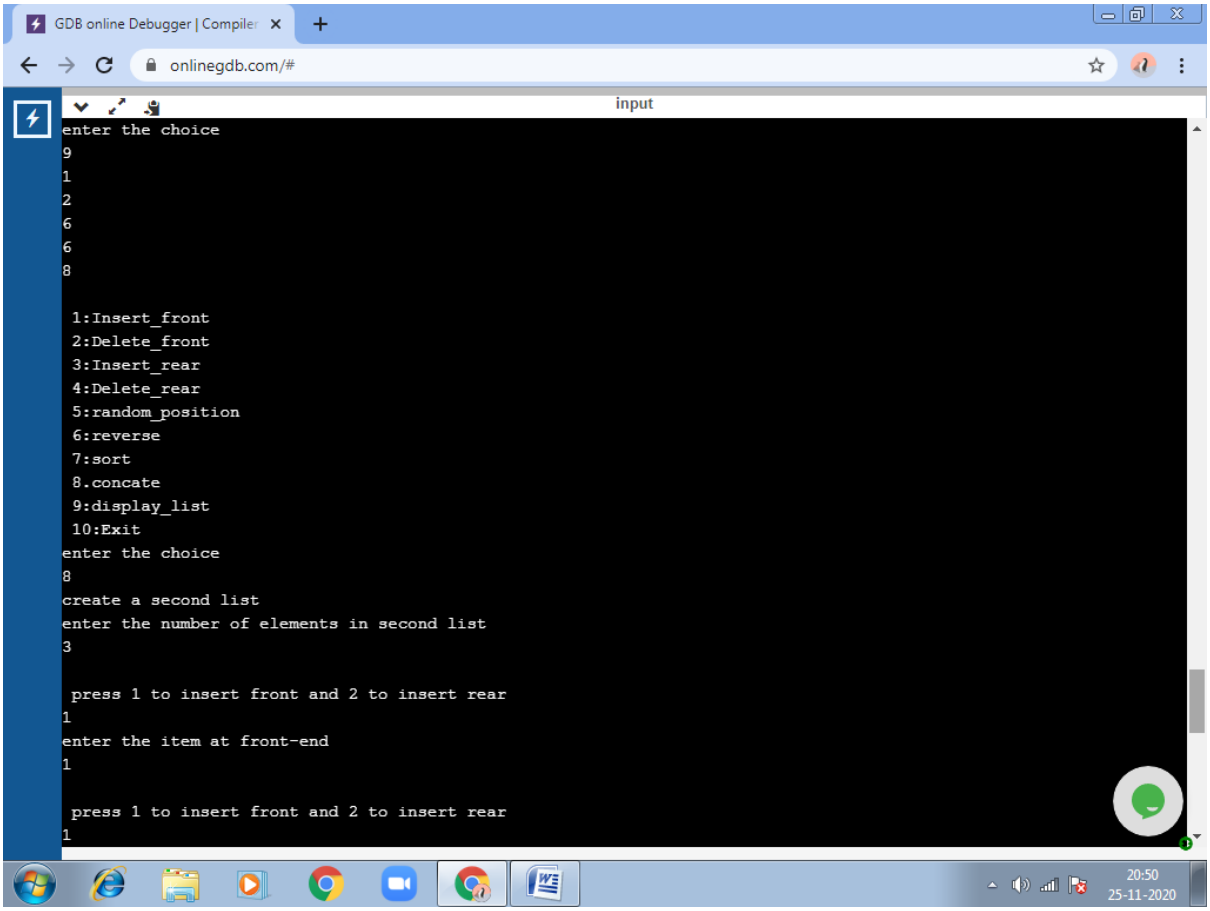

GDB online Debugger | Compiler
onlinegdb.com/#
input
enter the choice
9
2
1
1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:random_position
6:reverse
7:sort
8.concate
9:display_list
10:Exit
enter the choice
1
enter the item at front-end
6
1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:random_position
6:reverse
7:sort
8.concate
9:display_list
10:Exit
enter the choice
20:47
25-11-2020


GDB online Debugger | Compiler
onlinegdb.com/#
input
9:display_list
10:Exit
enter the choice
1
enter the item at front-end
8
1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:random_position
6:reverse
7:sort
8.concate
9:display_list
10:Exit
enter the choice
9
8
6
2
1
1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:random_position
6:reverse
7:sort
20:48
25-11-2020

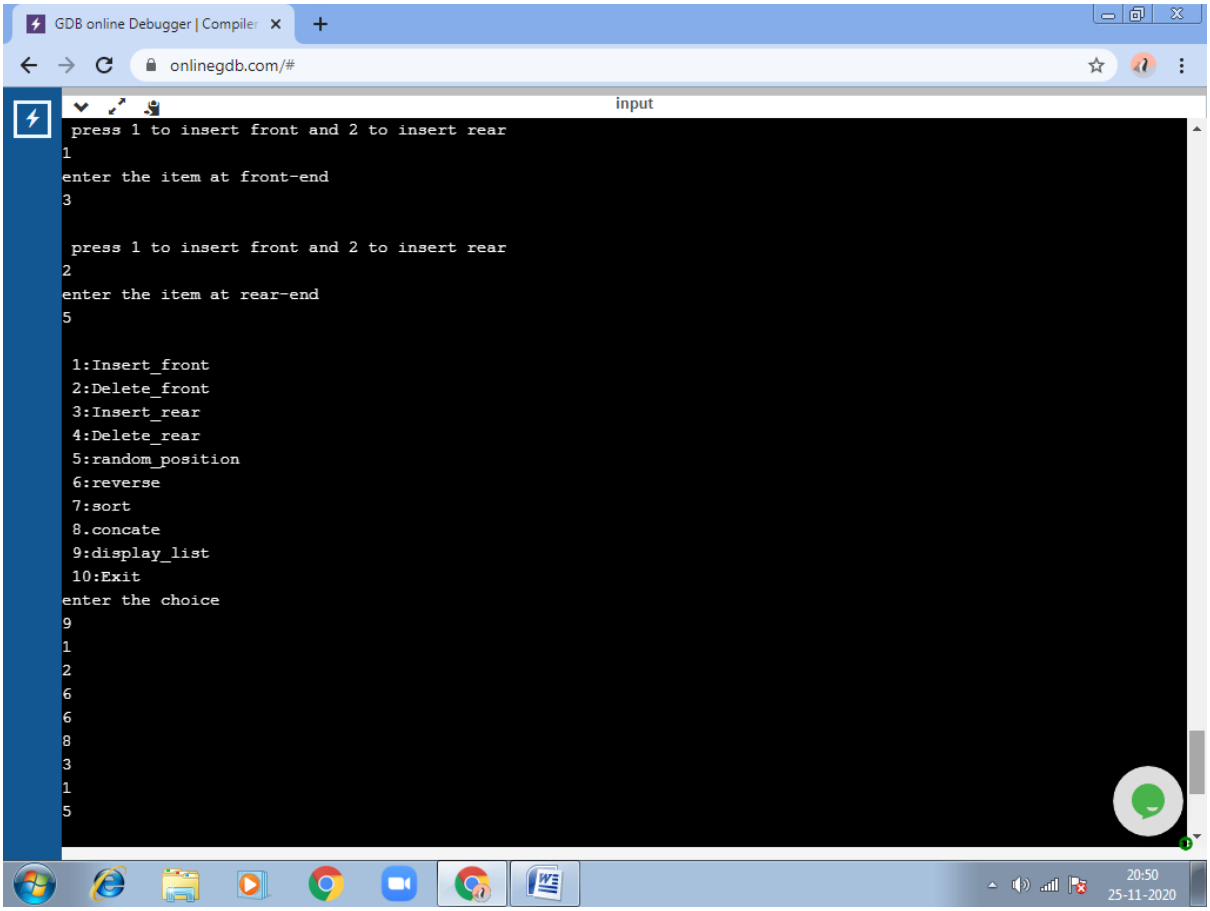

GDB online Debugger | Compiler
onlinegdb.com/#
input
7:sort
8.concate
9:display_list
10:Exit
enter the choice
5
press 1 to insert or 2 to delete at any desired position
1
enter the position to insert
3
enter the item to insert
6
1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:random_position
6:reverse
7:sort
8.concate
9:display_list
10:Exit
enter the choice
9
8
6
6
2
1
20:49
25-11-2020

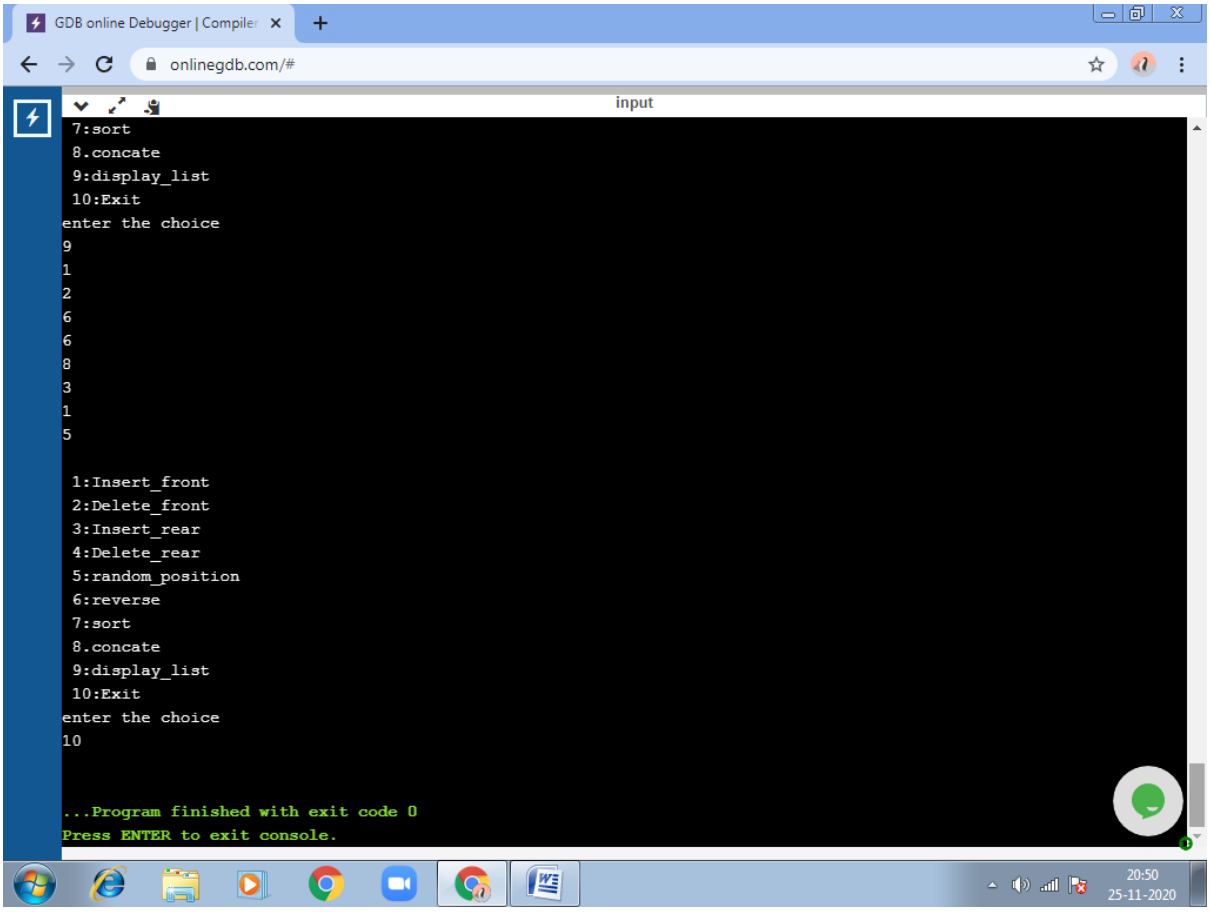

GDB online Debugger | Compiler
onlinegdb.com/#
input
2
1
1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:random_position
6:reverse
7:sort
8.concate
9:display_list
10:Exit
enter the choice
6
1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:random_position
6:reverse
7:sort
8.concate
9:display_list
10:Exit
enter the choice
9
1
2
6
20:49
25-11-2020


GDB online Debugger | Compiler
onlinegdb.com/#
input
6
8
1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:random_position
6:reverse
7:sort
8.concate
9:display_list
10:Exit
enter the choice
7
press 1 for ascending sort and 2 for descending sort:
1
1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:random_position
6:reverse
7:sort
8.concate
9:display_list
10:Exit
enter the choice
9
1
20:49
25-11-2020


GDB online Debugger | Compiler
onlinegdb.com/#
input
enter the choice
9
1
2
6
6
8
1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:random_position
6:reverse
7:sort
8.concate
9:display_list
10:Exit
enter the choice
8
create a second list
enter the number of elements in second list
3
press 1 to insert front and 2 to insert rear
1
enter the item at front-end
1
press 1 to insert front and 2 to insert rear
1
20:50
25-11-2020

\


GDB online Debugger | Compiler
onlinegdb.com/#
input
press 1 to insert front and 2 to insert rear
1
enter the item at front-end
3
press 1 to insert front and 2 to insert rear
2
enter the item at rear-end
5
1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:random_position
6:reverse
7:sort
8.concate
9:display_list
10:Exit
enter the choice
9
1
2
6
6
8
3
1
5
20:50
25-11-2020


GDB online Debugger | Compiler
onlinegdb.com/#
input
7:sort
8.concate
9:display_list
10:Exit
enter the choice
9
1
2
6
6
8
3
1
5
1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:random_position
6:reverse
7:sort
8.concate
9:display_list
10:Exit
enter the choice
10
...Program finished with exit code 0
Press ENTER to exit console.
20:50
25-11-2020

OBSERVATION:

```
#include<stdio.h>
#include<conio.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 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 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 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 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 delete_rear(NODE first)
{
NODE cur,prev;
if(first==NULL)
{
printf("list is empty cannot delete\n");
return first;
}
if(first->link==NULL)
{
printf("item deleted is %d\n",first->info);
free(first);
return NULL;
}
```

```
prev=NULL;
cur=first;
while(cur->link!=NULL)
{
prev=cur;
cur=cur->link;
}
printf("iten deleted at rear-end is %d",cur->info);
free(cur);
prev->link=NULL;
return first;
}
NODE insert_pos(int item,int pos,NODE first)
{
NODE temp;
NODE prev,cur;
int count;
temp=getnode();
temp->info=item;
temp->link=NULL;
if(first==NULL && pos==1)
return temp;
if(first==NULL)
{
 printf("invalid pos\n");
 return first;
}
```

```
if(pos==1)
{
temp->link=first;
return temp;
}
count=1;
prev=NULL;
cur=first;
while(cur!=NULL && count!=pos)
{
 prev=cur;
 cur=cur->link;
 count++;
}
if(count==pos)
{
prev->link=temp;
temp->link=cur;
return first;
}
printf("Invalid Position \n");
return first;
}
NODE delete_pos(int pos, NODE first)
{
NODE cur;
NODE prev;
```

```
int count;
if(first==NULL || pos<=0)
{
printf("invalid position \n");
return NULL;
}
if (pos==1)
{
cur=first;
 first=first->link;
freenode(cur);
return first;
}
prev=NULL;
cur=first;
count=1;
while(cur!=NULL)
{
if(count==pos)
break;  //if found
prev=cur;
cur=cur->link;
count++;
}
if(count!=pos)
{
  printf("invalid position\n");
```

```
   return first;
}
if(count!=pos)
{
  printf("invalid position specified\n");
  return first;
}

prev->link=cur->link;
 freenode(cur);
return first;
}
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 asc(NODE first)
{
```

```
   NODE prev=first;
   NODE cur=NULL;
                int temp;


if(first== NULL) {
             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 des(NODE first)
```

```
{
   NODE prev=first;
   NODE cur=NULL;
                int temp;

if(first==NULL) {
            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 concate(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)
{
 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);
  }
}
```

```
void main()
{
int item,choice,pos,element,option,choice2,item1,num;
NODE first=NULL;
NODE second=NULL;

for(;;)
{
printf("\n 1:Insert_front\n 2:Delete_front\n 3:Insert_rear\n 4:Delete_rear\n 5:random_position\n 6:reverse\n 7:sort\n 8.concate\n 9:display_list\n 10: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:printf("enter the item at rear-end\n");
            scanf("%d",&item);
            first=insert_rear(first,item);
            break;
  case 4:first=delete_rear(first);
            break;
 case 5:
 printf("press 1 to insert or 2 to delete at any desired position \n");
```

```
scanf("%d",&element);
if(element==1){
      printf("enter the position to insert \n");
                  scanf("%d",&pos);
                  printf("enter the item to insert \n");
                  scanf("%d",&item);
                  first=insert_pos(item,pos,first);
}
if(element==2){
      printf("enter the position to delete \n");
                  scanf("%d",&pos);
                  first=delete_pos(pos,first);
}
                  break;
 case 6:
     first=reverse(first);
     break;
 case 7:
     printf("press 1 for ascending sort and 2 for descending sort:\n");
     scanf("%d",&option);
     if(option==1)
     first=asc(first);
     if(option==2)
     first=des(first);
     break;
 case 8:
     printf("create a second list\n");
```

```
      printf("enter the number of elements in second list\n");

      scanf("%d",&num);
       for(int i=1;i<=num;i++){
      printf("\n press 1 to insert front and 2 to insert rear \n");
      scanf("%d",&choice2);

         if(choice2==1){
         printf("enter the item at front-end\n");
                  scanf("%d",&item1);
        second=IF(second,item1);
         }

        if(choice2==2){
        printf("enter the item at rear-end\n");
                 scanf("%d",&item1);
        second=IR(second,item1);
        }
       }

      first=concate(first,second);
      break;

 case 9:display(first);
            break;
default:exit(0);
            break;
```

```
 }

}

getch();}
```