UNIX ASSIGNMENT:4

NAME:JOSHUA STEPHEN

SEC:A

ROLL NUMBER:422148

CODE:

doublylinkedlist.h

```
#include <stdio.h>

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

int init(struct node **head, struct node **tail);
int insert(struct node **head, struct node **tail, int data, int
pos); int deletenode(struct node **head, struct node **tail,
int pos, int *data); int search(struct node **head, int key, int
*pos);
int traverseforward(struct node **head);
int traversebackward(struct node **tail);
int findsmallbig(struct node** head, int *big, int *small);
```

doublylinkedlist.c

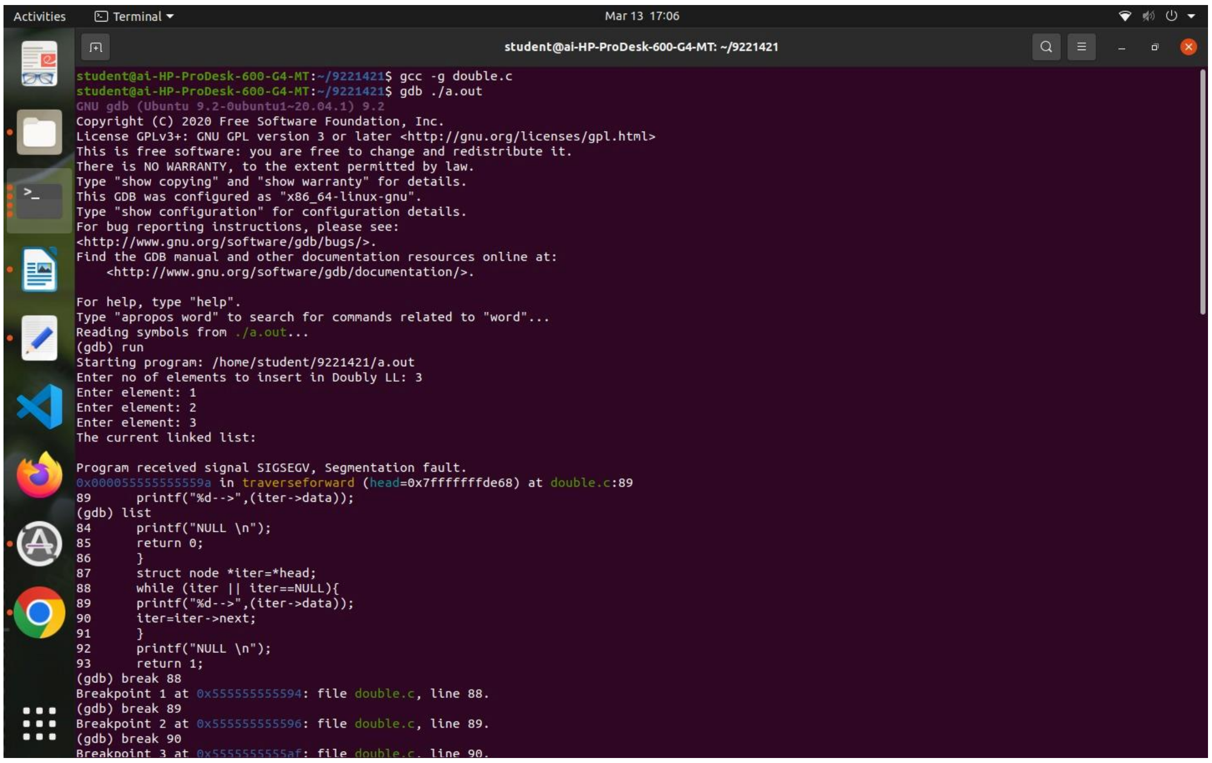
```
#include <stdio.h>
#include <stdlib.h>
#include "doublylinkedlist.h"



int init(struct node **head,struct node **tail){
        *head=NULL;
        *tail=NULL;
        return 1;
}

int insert(struct node **head, struct node **tail, int data, int pos){


        struct node *newnode=(struct node
        *)malloc(sizeof(struct node)); if (!newnode || pos<1)
        return 0;
        newnode->data=data;

        if(*head==NULL){
                if (pos==1){
                        newnode->next=NULL;
                        newnode->prev=NULL;
                        *head=newnode;
                        *tail=newnode;
                        return 1;
                }
                else return 0;
        }
        if (pos==1){
                (*head)->prev=newnode;
                newnode->next=*head;
                *head=newnode;
                newnode->prev=NULL;
                return 1;
        }

        struct node *ptr=NULL;
        ptr=*head;
        for(int i=1;i<pos-1 && ptr!=NULL;i++){
                ptr=ptr->next;
        }
        if (!ptr) return 0;
        newnode->next=ptr->next;
        newnode->prev=ptr;
        ptr->next=newnode;

        if ((newnode->next)==NULL) *tail=newnode;
        else (newnode->next)->prev=newnode;
        return 1;
}

int deletenode(struct node **head, struct node **tail, int

        pos, int *key){ if (*head==NULL || pos<1) return 0;


        struct node *iter=*head;
        int i=1;
        while (iter!=NULL && i<pos){
                iter=iter->next;
                i+=1;
        }
        if (!iter) return 0;
        *key=iter->data;
        if (iter==*head){
                *head=(*head)->next;
                (*head)->prev=NULL;
                free(iter);
                return 1;
        }
        if (iter==*tail){
                *tail=(*tail)->prev;
                (*tail)->next=NULL;
                free(iter);
                return 1;
        }
        (iter->next)->prev=iter->prev;
        (iter->prev)->next=iter->next;
        free(iter);
        return 1;

}

int search(struct node **head, int key,
 int *pos){ if (*head==NULL) return 0;

        struct node *iter=*head;
        int i=1;
            while (iter!=NULL && iter-
          >data!=key){ iter=iter->next;
                i+=1;
        }
        if (iter==NULL) return 0;
        *pos=i;
        return 1;
}
int traverseforward(struct node **head){
        if (*head==NULL){
                printf("NULL \n");
                return 0;
        }
        struct node *iter=*head;
        while (iter){
                printf("%d-->",(iter->data));
                iter=iter->next;
        }
        printf("NULL \n");
        return 1;
}

int traversebackward(struct node **tail){
        if (*tail==NULL){
                printf("NULL \n");
                return 0;
        }
        struct node *iter=*tail;
        while (iter){
                printf("%d-->",(iter->data));
                iter=iter->prev;
        }
        printf("NULL \n");
        return 1;
}
```

```
int findsmallbig(struct node **head, int *big, int
        *small){ if (*head==NULL) return 0;
        struct node *iter=*head;
        int tempsmall=(*head)->data;
        int tempbig=tempsmall;

        while (iter!=NULL){
                if (tempbig<(iter->data)) tempbig=iter->data;
                if (tempsmall>(iter->data))
                tempsmall=iter->data; iter=iter->next;
        }
        *big=tempbig;
        *small=tempsmall;
        return 1;
}

int main(){
        struct node *head=NULL;
        struct node *tail=NULL;
        init(&head, &tail);

        int length;
        printf("Enter no of elements to insert in
        Doubly LL: "); scanf("%d",&length);
        for(int i=1; i<=length;i++){
                int elem;
                printf("Enter element: ");
                scanf("%d",&elem);
                insert(&head,&tail,elem,i);
        }
```

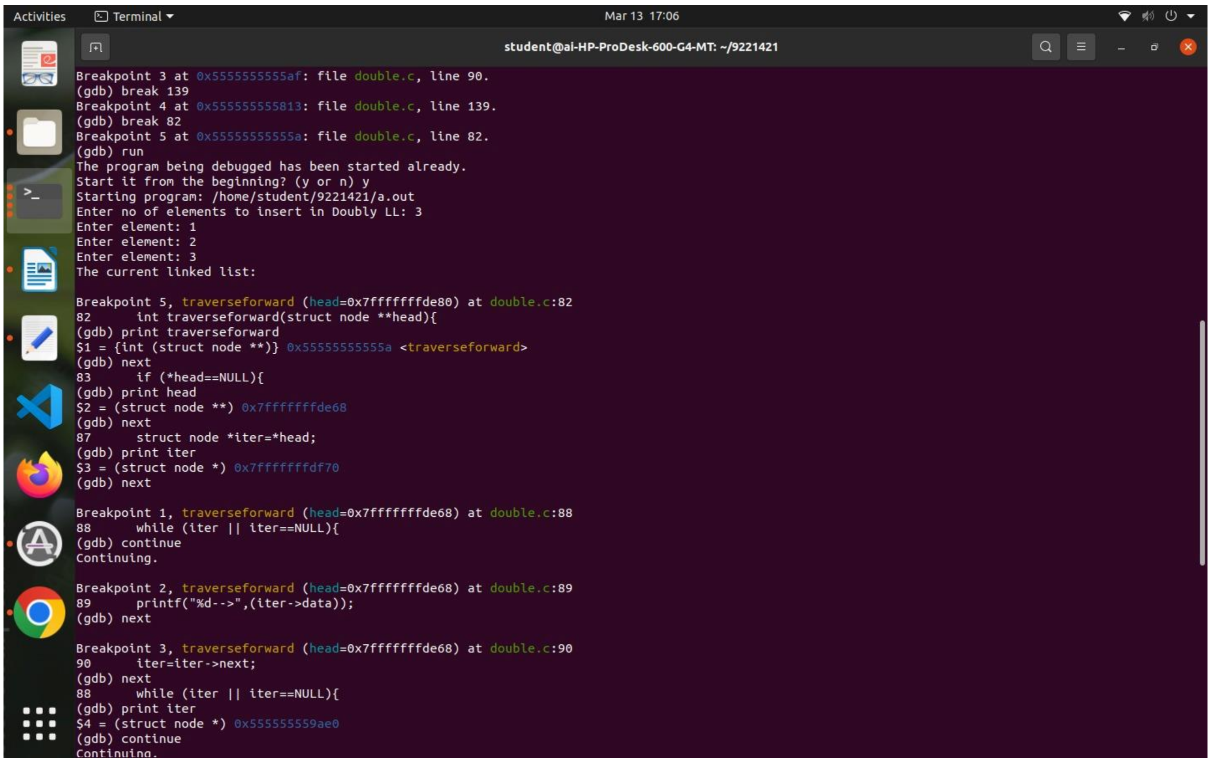
```
printf("The current linked list: \n");
traverseforward(&head);

int elem,pos;
printf("Enter element to insert at specific position: ");
scanf("%d %d",&elem,&pos);
insert(&head,&tail,elem,pos);
printf("The current linked list: \n");
traverseforward(&head);

printf("Traversing in backward direction: \n");
traversebackward(&tail);

printf("Deleting element: \n");

printf("Enter position of element to delete: ");
scanf("%d",&pos);
deletenode(&head, &tail,pos,&elem);
printf("The current linked list: \n");
traverseforward(&head);
printf("Deleted element: %d \n",elem);

printf("Enter element to search: ");
scanf("%d",&elem);
search(&head, elem, &pos);
printf("Position of element: %d \n", pos);

int big,small;
findsmallbig(&head,&big,&small);
printf("The larget and smallest elements are: %d %d \n",
big, small); return 0;
```

}

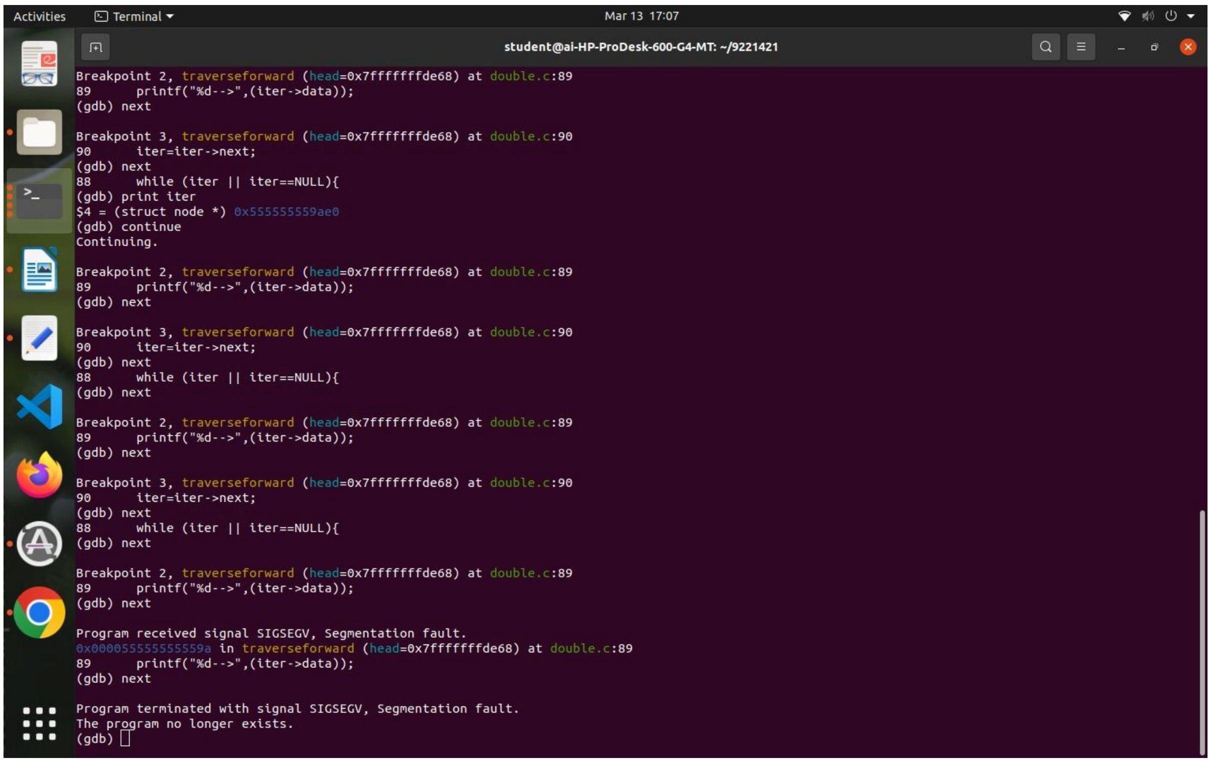
Output:



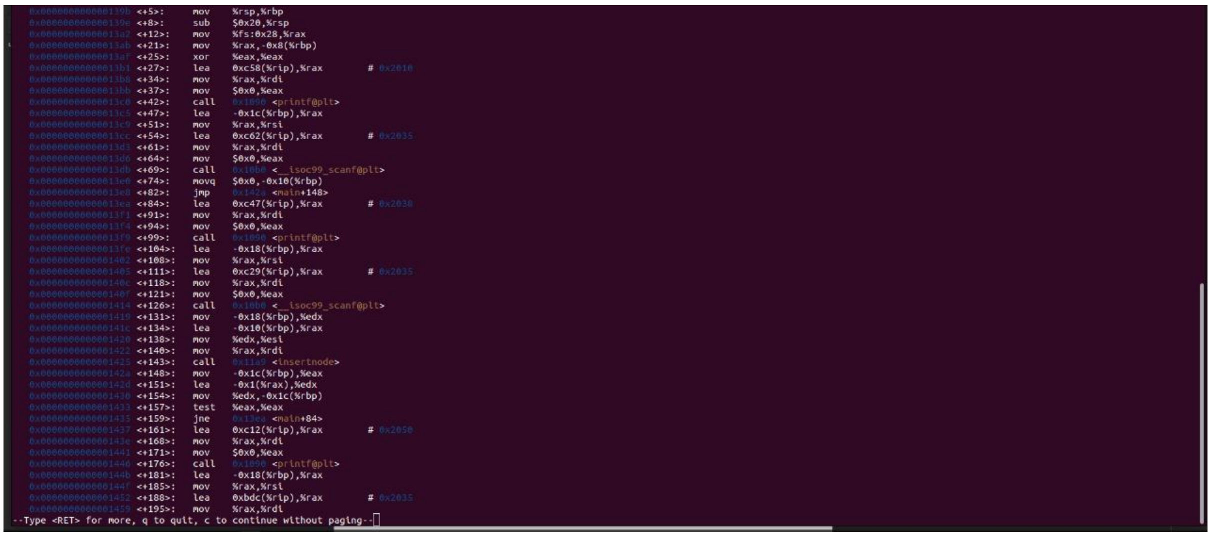
```
student@ai-HP-ProDesk-600-G4-MT:~/9221421$ gcc -g double.c
student@ai-HP-ProDesk-600-G4-MT:~/9221421$ gdb ./a.out
GNU gdb (Ubuntu 9.2-0ubuntu1~20.04.1) 9.2
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./a.out...
(gdb) run
Starting program: /home/student/9221421/a.out
Enter no of elements to insert in Doubly LL: 3
Enter element: 1
Enter element: 2
Enter element: 3
The current linked list:

Program received signal SIGSEGV, Segmentation fault.
0x000055555555559a in traverseforward (head=0x7fffffffde68) at double.c:89
89        printf("%d-->",(iter->data));
(gdb) list
84        printf("NULL \n");
85        return 0;
86        }
87        struct node *iter=*head;
88        while (iter || iter==NULL){
89        printf("%d-->",(iter->data));
90        iter=iter->next;
91        }
92        printf("NULL \n");
93        return 1;
(gdb) break 88
Breakpoint 1 at 0x555555555594: file double.c, line 88.
(gdb) break 89
Breakpoint 2 at 0x555555555596: file double.c, line 89.
(gdb) break 90
Breakpoint 3 at 0x5555555555af: file double.c, line 90.
```



```
Breakpoint 3 at 0x5555555555af: file double.c, line 90.
(gdb) break 139
Breakpoint 4 at 0x555555555813: file double.c, line 139.
(gdb) break 82
Breakpoint 5 at 0x55555555555a: file double.c, line 82.
(gdb) run
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/student/9221421/a.out
Enter no of elements to insert in Doubly LL: 3
Enter element: 1
Enter element: 2
Enter element: 3
The current linked list:

Breakpoint 5, traverseforward (head=0x7fffffffde80) at double.c:82
82        int traverseforward(struct node **head){
(gdb) print traverseforward
$1 = {int (struct node **)} 0x55555555555a <traverseforward>
(gdb) next
83        if (*head==NULL){
(gdb) print head
$2 = (struct node **) 0x7fffffffde68
(gdb) next
87        struct node *iter=*head;
(gdb) print iter
$3 = (struct node *) 0x7fffffffdf70
(gdb) next

Breakpoint 1, traverseforward (head=0x7fffffffde68) at double.c:88
88        while (iter || iter==NULL){
(gdb) continue
Continuing.

Breakpoint 2, traverseforward (head=0x7fffffffde68) at double.c:89
89        printf("%d-->",(iter->data));
(gdb) next

Breakpoint 3, traverseforward (head=0x7fffffffde68) at double.c:90
90        iter=iter->next;
(gdb) next
88        while (iter || iter==NULL){
(gdb) print iter
$4 = (struct node *) 0x555555559ae0
(gdb) continue
Continuing.
```



```
Breakpoint 2, traverseforward (head=0x7fffffffde68) at double.c:89
89      printf("%d-->",(iter->data));
(gdb) next

Breakpoint 3, traverseforward (head=0x7fffffffde68) at double.c:90
90      iter=iter->next;
(gdb) next
88      while (iter || iter==NULL){
(gdb) print iter
$4 = (struct node *) 0x555555559ae0
(gdb) continue
Continuing.

Breakpoint 2, traverseforward (head=0x7fffffffde68) at double.c:89
89      printf("%d-->",(iter->data));
(gdb) next

Breakpoint 3, traverseforward (head=0x7fffffffde68) at double.c:90
90      iter=iter->next;
(gdb) next
88      while (iter || iter==NULL){
(gdb) next

Breakpoint 2, traverseforward (head=0x7fffffffde68) at double.c:89
89      printf("%d-->",(iter->data));
(gdb) next

Breakpoint 3, traverseforward (head=0x7fffffffde68) at double.c:90
90      iter=iter->next;
(gdb) next
88      while (iter || iter==NULL){
(gdb) next

Breakpoint 2, traverseforward (head=0x7fffffffde68) at double.c:89
89      printf("%d-->",(iter->data));
(gdb) next

Program received signal SIGSEGV, Segmentation fault.
0x000055555555559a in traverseforward (head=0x7fffffffde68) at double.c:89
89      printf("%d-->",(iter->data));
(gdb) next

Program terminated with signal SIGSEGV, Segmentation fault.
The program no longer exists.
(gdb) 
```



```
   0x000000000000139b <+5>:     mov    %rsp,%rbp
   0x000000000000139e <+8>:     sub    $0x20,%rsp
   0x00000000000013a2 <+12>:    mov    %fs:0x28,%rax
   0x00000000000013ab <+21>:    mov    %rax,-0x8(%rbp)
   0x00000000000013af <+25>:    xor    %eax,%eax
   0x00000000000013b1 <+27>:    lea    0xc58(%rip),%rax        # 0x2010
   0x00000000000013b8 <+34>:    mov    %rax,%rdi
   0x00000000000013bb <+37>:    mov    $0x0,%eax
   0x00000000000013c0 <+42>:    call   0x1090 <printf@plt>
   0x00000000000013c5 <+47>:    lea    -0x1c(%rbp),%rax
   0x00000000000013c9 <+51>:    mov    %rax,%rsi
   0x00000000000013cc <+54>:    lea    0xc62(%rip),%rax        # 0x2035
   0x00000000000013d3 <+61>:    mov    %rax,%rdi
   0x00000000000013d6 <+64>:    mov    $0x0,%eax
   0x00000000000013db <+69>:    call   0x10b0 <__isoc99_scanf@plt>
   0x00000000000013e0 <+74>:    movq   $0x0,-0x10(%rbp)
   0x00000000000013e8 <+82>:    jmp    0x142a <main+148>
   0x00000000000013ea <+84>:    lea    0xc47(%rip),%rax        # 0x2038
   0x00000000000013f1 <+91>:    mov    %rax,%rdi
   0x00000000000013f4 <+94>:    mov    $0x0,%eax
   0x00000000000013f9 <+99>:    call   0x1090 <printf@plt>
   0x00000000000013fe <+104>:   lea    -0x18(%rbp),%rax
   0x0000000000001402 <+108>:   mov    %rax,%rsi
   0x0000000000001405 <+111>:   lea    0xc29(%rip),%rax        # 0x2035
   0x000000000000140c <+118>:   mov    %rax,%rdi
   0x000000000000140f <+121>:   mov    $0x0,%eax
   0x0000000000001414 <+126>:   call   0x10b0 <__isoc99_scanf@plt>
   0x0000000000001419 <+131>:   mov    -0x18(%rbp),%edx
   0x000000000000141c <+134>:   lea    -0x10(%rbp),%rax
   0x0000000000001420 <+138>:   mov    %edx,%esi
   0x0000000000001422 <+140>:   mov    %rax,%rdi
   0x0000000000001425 <+143>:   call   0x11a9 <insertnode>
   0x000000000000142a <+148>:   mov    -0x1c(%rbp),%eax
   0x000000000000142d <+151>:   lea    -0x1(%rax),%edx
   0x0000000000001430 <+154>:   mov    %edx,-0x1c(%rbp)
   0x0000000000001433 <+157>:   test   %eax,%eax
   0x0000000000001435 <+159>:   jne    0x13ea <main+84>
   0x0000000000001437 <+161>:   lea    0xc12(%rip),%rax        # 0x2050
   0x000000000000143e <+168>:   mov    %rax,%rdi
   0x0000000000001441 <+171>:   mov    $0x0,%eax
   0x0000000000001446 <+176>:   call   0x1090 <printf@plt>
   0x000000000000144b <+181>:   lea    -0x18(%rbp),%rax
   0x000000000000144f <+185>:   mov    %rax,%rsi
   0x0000000000001452 <+188>:   lea    0xbdc(%rip),%rax        # 0x2035
   0x0000000000001459 <+195>:   mov    %rax,%rdi
--Type <RET> for more, q to quit, c to continue without paging--
```

Code:

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

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


struct node *head;

int initList(struct node **head){
	*head=NULL;
	return 1;
}


int search(struct node **head, int data, struct node
**ptrToKey, int *pos){
	if (*head==NULL) return 0;
	*pos=1;
	struct node *ptr=*head;
	for (;ptr!=NULL && ptr->data!=data;ptr=ptr->next){
		*pos=(*pos)+1;
	}
		*ptrToKey=ptr;

	if (!ptr) return 0;
	return 1;
}


int insert(struct node **head, int position, int data){
	struct node *newnode=(struct node
*)malloc(sizeof(struct node));
	if (newnode==NULL) return 0;

	newnode->data=data;
	if (position==1){
		newnode->next=*head;
		*head = newnode;
		return 1;
	}
```

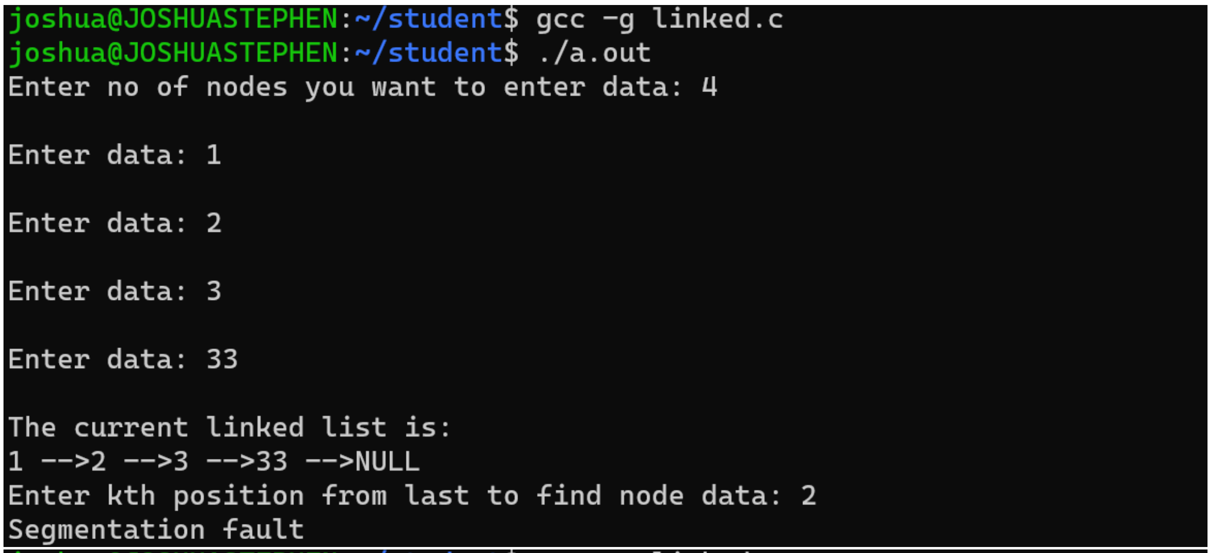
```
}

	//to make sure there are no duplicate insertions we
search if given data is already present in linked list

	struct node *ptrToKey=NULL;

	int pos=0;

	if (!search(head, data,&ptrToKey, &pos)){

		struct node *ptr=*head;

		for (int i=1; i<position-1 && ptr!=NULL;i++)

			ptr=ptr->next;


		if (ptr==NULL) return 0;

		else{

			newnode->next=ptr->next;

			ptr->next=newnode;

			return 1;

		}

	}

	else{

		printf("Element already present in address: %p
\n",ptrToKey);

		return 0;

	}

}
```

```
int traverse(struct node *head){
        if (!head){
                printf("NULL \n");
                return 1;
        }

        for (struct node *ptr=head;ptr!=NULL;ptr=ptr->next)
                printf("%d -->",ptr->data);
        printf("NULL \n");
        return 1;
}

int kFromLast(struct node *head, int k ,int *data){
        if(!head) return 0;
        struct node *fast=head;
        struct node *slow=NULL;
        int i=1;

        while(fast!=NULL && i<=k){
                fast=fast->next;
                i++;
```

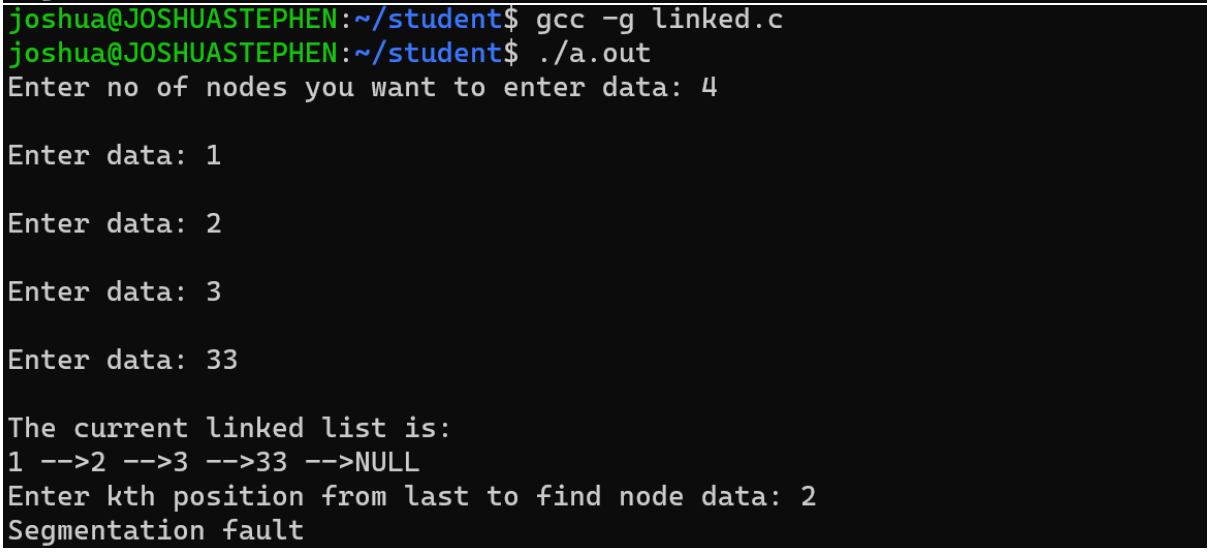
```
	}
	if(fast==NULL && i<k) return 0;
	slow=head;
	while(slow!=NULL){
		slow=slow->next;
		fast=fast->next;
	}
	*data=slow->data;
	return 1;
}

int main(){
	struct node *head;
	initList(&head);
	int n;

	printf("Enter no of nodes you want to enter data: ");
	scanf("%d",&n);
	int pos=1;
	while (n--){
		int data;
		printf("\nEnter data: ");
```

```
                scanf("%d",&data);
                if (!insert(&head,pos++,data)) return 0;
        }

        printf("\nThe current linked list is:\n");
        traverse(head);

        int k, data;
        printf("Enter kth position from last to find node data: ");
        scanf("%d", &k);

        kFromLast(head, k, &data);
        printf("Data: %d\n",data);
        return 0;
}
```

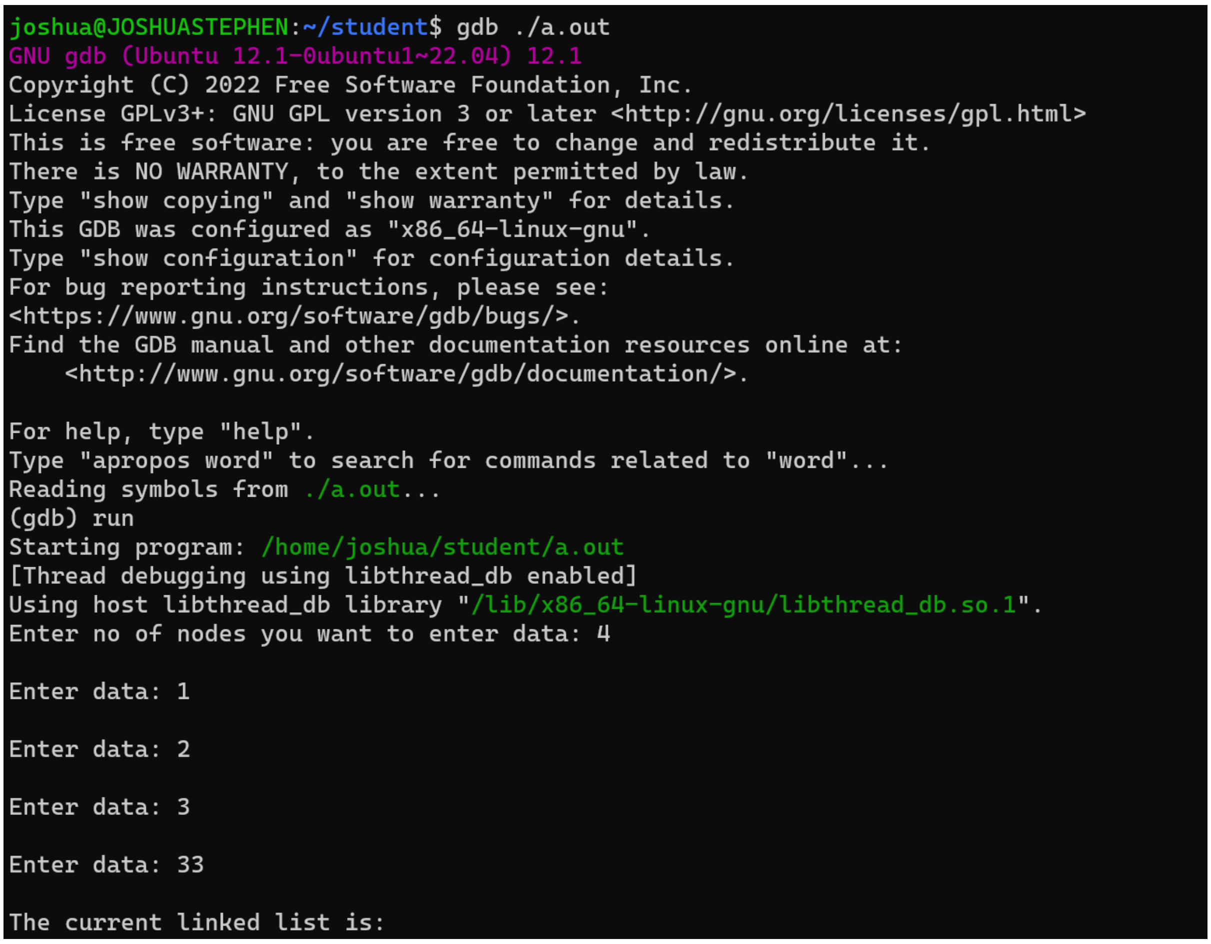
Output:



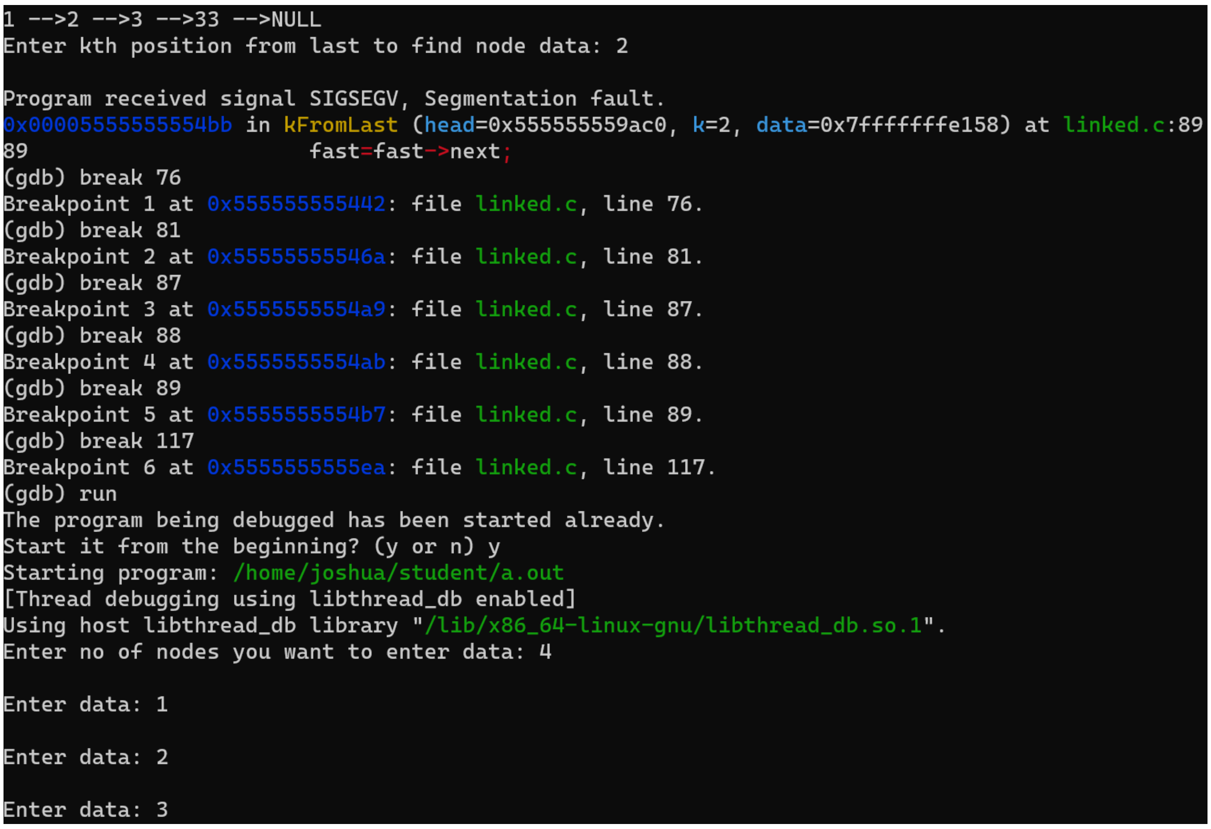
```
joshua@JOSHUASTEPHEN:~/student$ gcc -g linked.c
joshua@JOSHUASTEPHEN:~/student$ ./a.out
Enter no of nodes you want to enter data: 4

Enter data: 1

Enter data: 2

Enter data: 3

Enter data: 33

The current linked list is:
1 -->2 -->3 -->33 -->NULL
Enter kth position from last to find node data: 2
Segmentation fault
```



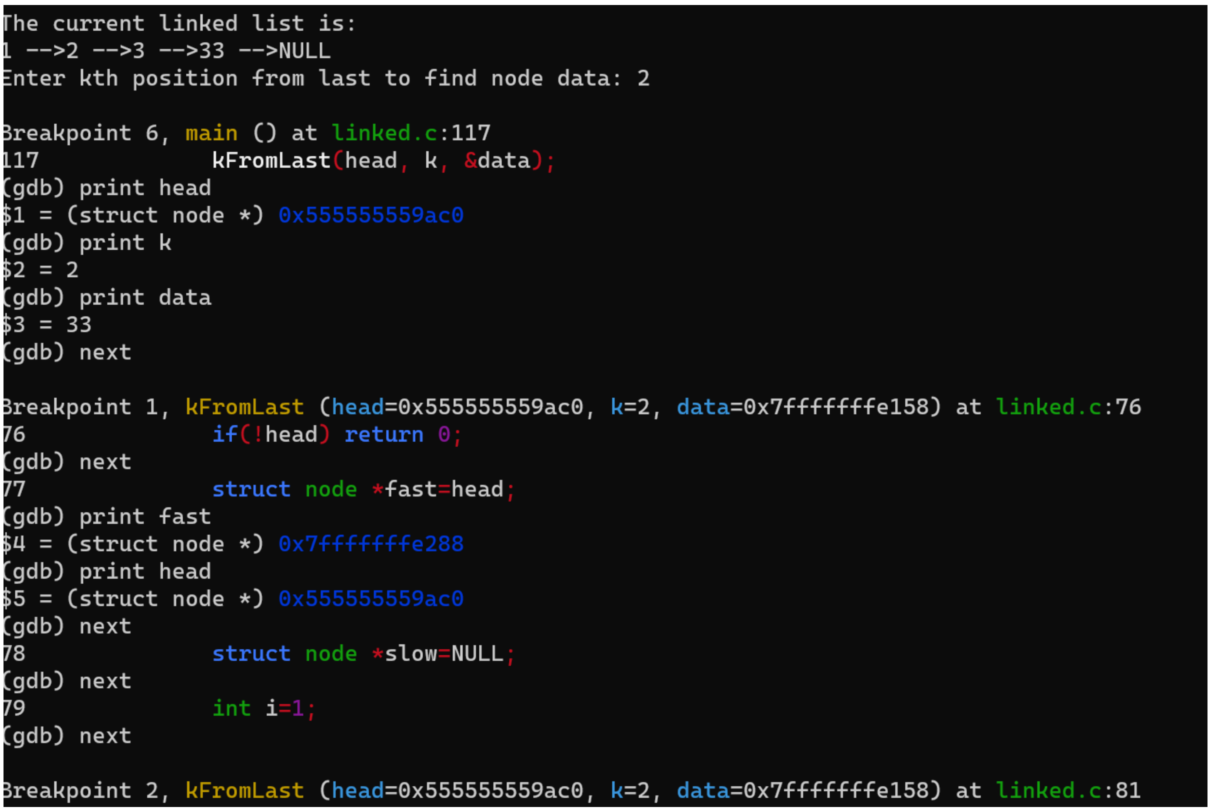
```
joshua@JOSHUASTEPHEN:~/student$ gcc -g linked.c
joshua@JOSHUASTEPHEN:~/student$ ./a.out
Enter no of nodes you want to enter data: 4

Enter data: 1

Enter data: 2

Enter data: 3

Enter data: 33

The current linked list is:
1 -->2 -->3 -->33 -->NULL
Enter kth position from last to find node data: 2
Segmentation fault
```



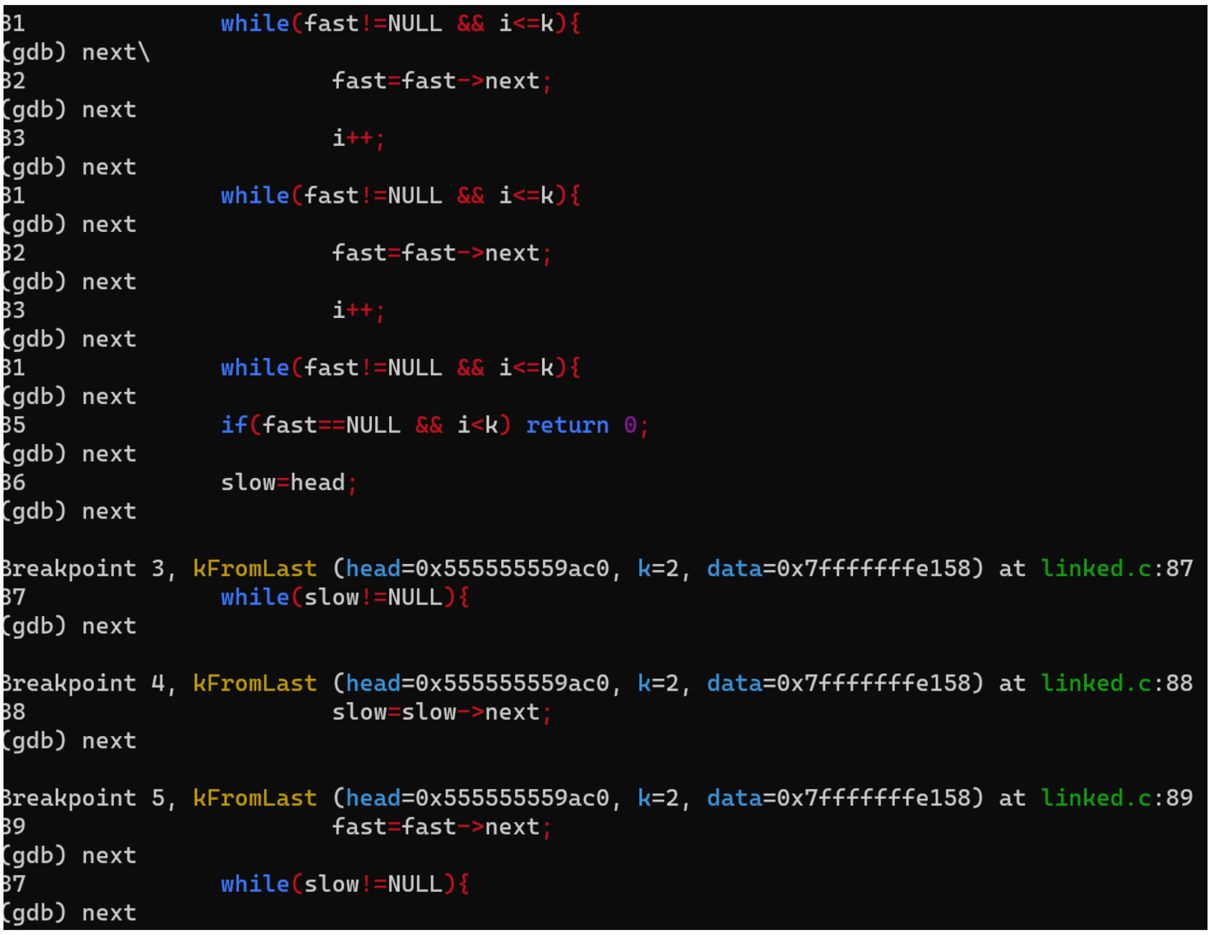
```
joshua@JOSHUASTEPHEN:~/student$ gdb ./a.out
GNU gdb (Ubuntu 12.1-0ubuntu1~22.04) 12.1
Copyright (C) 2022 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./a.out...
(gdb) run
Starting program: /home/joshua/student/a.out
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
Enter no of nodes you want to enter data: 4

Enter data: 1

Enter data: 2

Enter data: 3

Enter data: 33

The current linked list is:
```



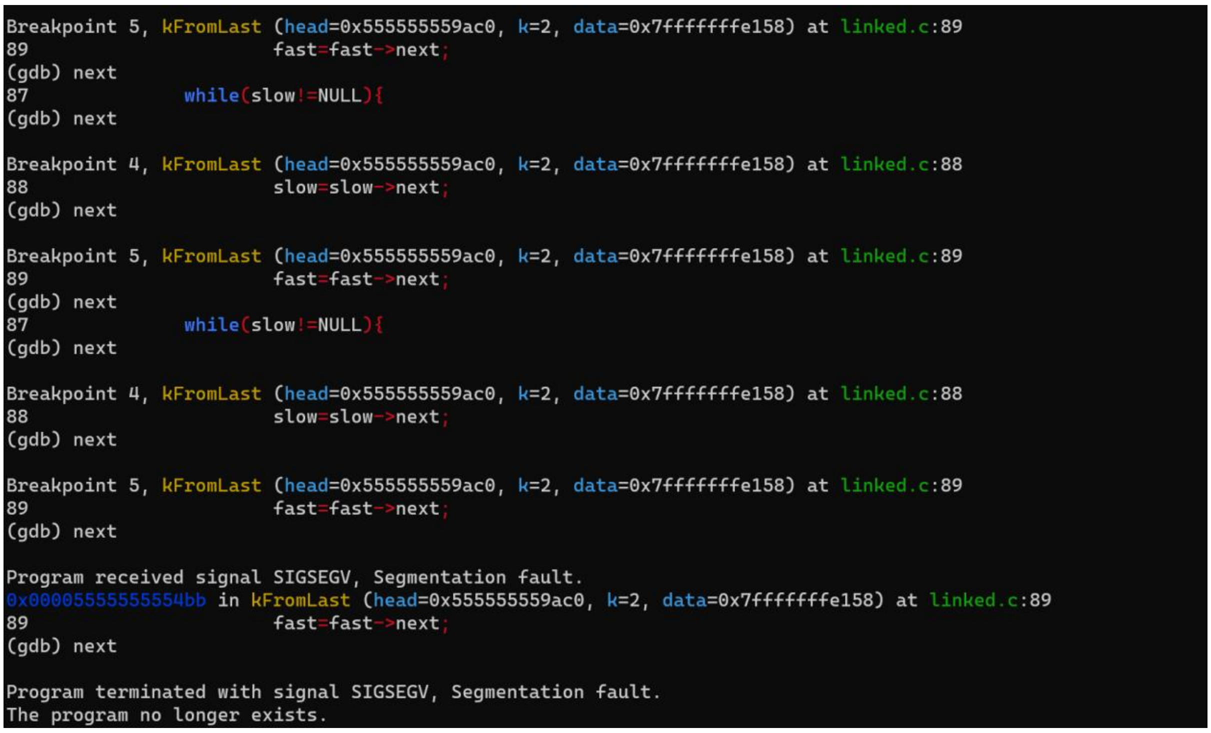
```
1 -->2 -->3 -->33 -->NULL
Enter kth position from last to find node data: 2

Program received signal SIGSEGV, Segmentation fault.
0x00005555555554bb in kFromLast (head=0x555555559ac0, k=2, data=0x7fffffffe158) at linked.c:89
89                              fast=fast->next;
(gdb) break 76
Breakpoint 1 at 0x555555555442: file linked.c, line 76.
(gdb) break 81
Breakpoint 2 at 0x55555555546a: file linked.c, line 81.
(gdb) break 87
Breakpoint 3 at 0x5555555554a9: file linked.c, line 87.
(gdb) break 88
Breakpoint 4 at 0x5555555554ab: file linked.c, line 88.
(gdb) break 89
Breakpoint 5 at 0x5555555554b7: file linked.c, line 89.
(gdb) break 117
Breakpoint 6 at 0x5555555555ea: file linked.c, line 117.
(gdb) run
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/joshua/student/a.out
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
Enter no of nodes you want to enter data: 4

Enter data: 1

Enter data: 2

Enter data: 3
```



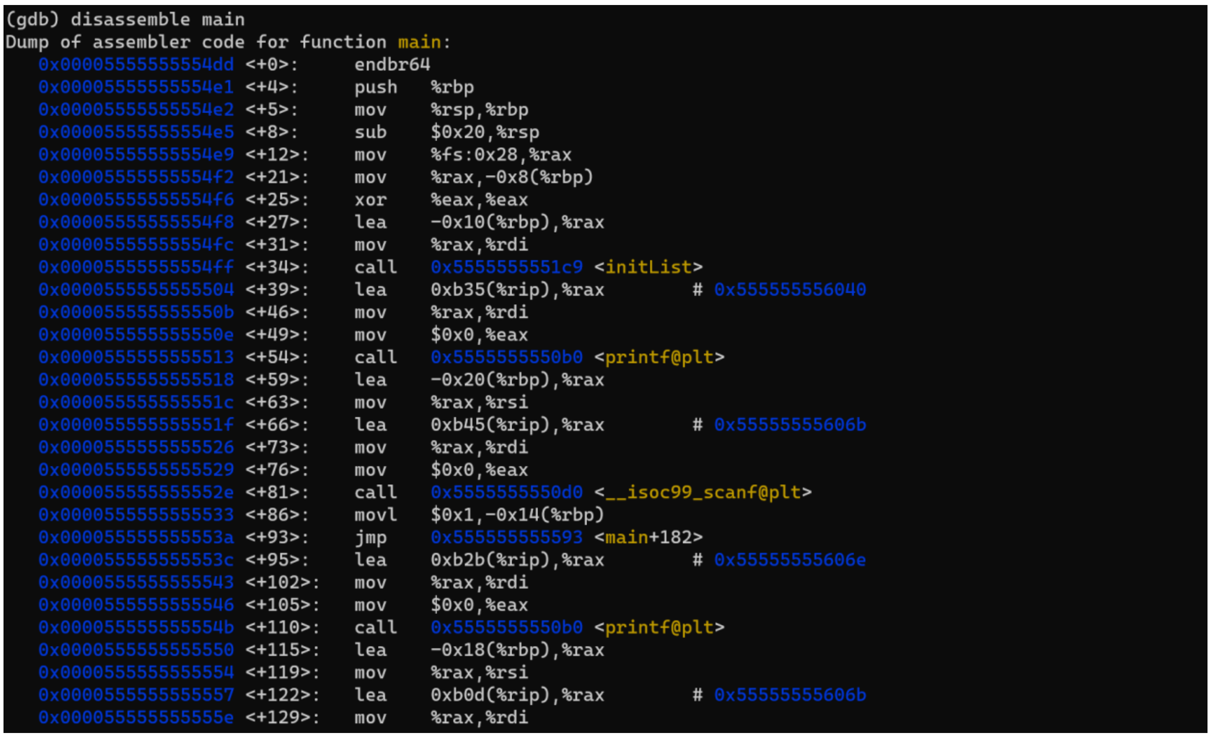
```
The current linked list is:
1 -->2 -->3 -->33 -->NULL
Enter kth position from last to find node data: 2

Breakpoint 6, main () at linked.c:117
117             kFromLast(head, k, &data);
(gdb) print head
$1 = (struct node *) 0x555555559ac0
(gdb) print k
$2 = 2
(gdb) print data
$3 = 33
(gdb) next

Breakpoint 1, kFromLast (head=0x555555559ac0, k=2, data=0x7fffffffe158) at linked.c:76
76              if(!head) return 0;
(gdb) next
77              struct node *fast=head;
(gdb) print fast
$4 = (struct node *) 0x7fffffffe288
(gdb) print head
$5 = (struct node *) 0x555555559ac0
(gdb) next
78              struct node *slow=NULL;
(gdb) next
79              int i=1;
(gdb) next

Breakpoint 2, kFromLast (head=0x555555559ac0, k=2, data=0x7fffffffe158) at linked.c:81
```



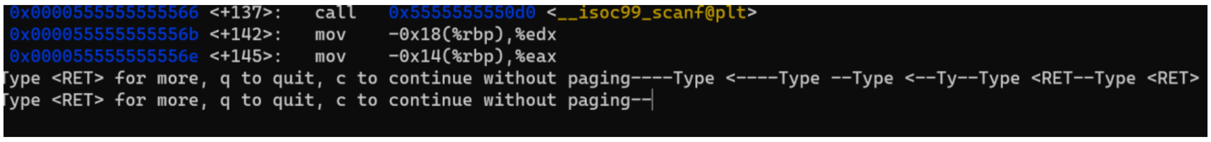
```
81              while(fast!=NULL && i<=k){
(gdb) next\
82                      fast=fast->next;
(gdb) next
83                      i++;
(gdb) next
81              while(fast!=NULL && i<=k){
(gdb) next
82                      fast=fast->next;
(gdb) next
83                      i++;
(gdb) next
81              while(fast!=NULL && i<=k){
(gdb) next
85              if(fast==NULL && i<k) return 0;
(gdb) next
86              slow=head;
(gdb) next

Breakpoint 3, kFromLast (head=0x555555559ac0, k=2, data=0x7fffffffe158) at linked.c:87
87              while(slow!=NULL){
(gdb) next

Breakpoint 4, kFromLast (head=0x555555559ac0, k=2, data=0x7fffffffe158) at linked.c:88
88                      slow=slow->next;
(gdb) next

Breakpoint 5, kFromLast (head=0x555555559ac0, k=2, data=0x7fffffffe158) at linked.c:89
89                      fast=fast->next;
(gdb) next
87              while(slow!=NULL){
(gdb) next
```



```
Breakpoint 5, kFromLast (head=0x555555559ac0, k=2, data=0x7fffffffe158) at linked.c:89
89                      fast=fast->next;
(gdb) next
87              while(slow!=NULL){
(gdb) next

Breakpoint 4, kFromLast (head=0x555555559ac0, k=2, data=0x7fffffffe158) at linked.c:88
88                      slow=slow->next;
(gdb) next

Breakpoint 5, kFromLast (head=0x555555559ac0, k=2, data=0x7fffffffe158) at linked.c:89
89                      fast=fast->next;
(gdb) next
87              while(slow!=NULL){
(gdb) next

Breakpoint 4, kFromLast (head=0x555555559ac0, k=2, data=0x7fffffffe158) at linked.c:88
88                      slow=slow->next;
(gdb) next

Breakpoint 5, kFromLast (head=0x555555559ac0, k=2, data=0x7fffffffe158) at linked.c:89
89                      fast=fast->next;
(gdb) next

Program received signal SIGSEGV, Segmentation fault.
0x00005555555554bb in kFromLast (head=0x555555559ac0, k=2, data=0x7fffffffe158) at linked.c:89
89                      fast=fast->next;
(gdb) next

Program terminated with signal SIGSEGV, Segmentation fault.
The program no longer exists.
```



```
(gdb) disassemble main
Dump of assembler code for function main:
   0x00005555555554dd <+0>:     endbr64
   0x00005555555554e1 <+4>:     push   %rbp
   0x00005555555554e2 <+5>:     mov    %rsp,%rbp
   0x00005555555554e5 <+8>:     sub    $0x20,%rsp
   0x00005555555554e9 <+12>:    mov    %fs:0x28,%rax
   0x00005555555554f2 <+21>:    mov    %rax,-0x8(%rbp)
   0x00005555555554f6 <+25>:    xor    %eax,%eax
   0x00005555555554f8 <+27>:    lea    -0x10(%rbp),%rax
   0x00005555555554fc <+31>:    mov    %rax,%rdi
   0x00005555555554ff <+34>:    call   0x5555555551c9 <initList>
   0x0000555555555504 <+39>:    lea    0xb35(%rip),%rax        # 0x555555556040
   0x000055555555550b <+46>:    mov    %rax,%rdi
   0x000055555555550e <+49>:    mov    $0x0,%eax
   0x0000555555555513 <+54>:    call   0x5555555550b0 <printf@plt>
   0x0000555555555518 <+59>:    lea    -0x20(%rbp),%rax
   0x000055555555551c <+63>:    mov    %rax,%rsi
   0x000055555555551f <+66>:    lea    0xb45(%rip),%rax        # 0x55555555606b
   0x0000555555555526 <+73>:    mov    %rax,%rdi
   0x0000555555555529 <+76>:    mov    $0x0,%eax
   0x000055555555552e <+81>:    call   0x5555555550d0 <__isoc99_scanf@plt>
   0x0000555555555533 <+86>:    movl   $0x1,-0x14(%rbp)
   0x000055555555553a <+93>:    jmp    0x555555555593 <main+182>
   0x000055555555553c <+95>:    lea    0xb2b(%rip),%rax        # 0x55555555606e
   0x0000555555555543 <+102>:   mov    %rax,%rdi
   0x0000555555555546 <+105>:   mov    $0x0,%eax
   0x000055555555554b <+110>:   call   0x5555555550b0 <printf@plt>
   0x0000555555555550 <+115>:   lea    -0x18(%rbp),%rax
   0x0000555555555554 <+119>:   mov    %rax,%rsi
   0x0000555555555557 <+122>:   lea    0xb0d(%rip),%rax        # 0x55555555606b
   0x000055555555555e <+129>:   mov    %rax,%rdi
```



```
 0x0000555555555566 <+137>:    call   0x5555555550d0 <__isoc99_scanf@plt>
 0x000055555555556b <+142>:    mov    -0x18(%rbp),%edx
 0x000055555555556e <+145>:    mov    -0x14(%rbp),%eax
Type <RET> for more, q to quit, c to continue without paging----Type <----Type --Type <--Ty--Type <RET--Type <RET>
Type <RET> for more, q to quit, c to continue without paging--
```