circular single linked list

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
 int data;
 struct node *link;
};
struct node *head = NULL, *x, *y, *z;
void create();
void ins_at_beg();
void ins_at_pos();
void del_at_beg();
void del_at_pos();
void traverse();
void search();
int main()
{
int ch;
printf("\n 1.Creation \n 2.Insertion at beginning \n 3.Insertion at remaining");
 printf("\n4.Deletion at beginning \n5.Deletion at remaining \n6.traverse\n7.search\n8.exit");
 while (1)
```

```
{
printf("\n Enter your choice:");
scanf("%d", &ch);
switch(ch)
{
case 1:
create();
break;
case 2:
ins_at_beg();
break;
case 3:
ins_at_pos();
break;
case 4:
del_at_beg();
break;
case 5:
del_at_pos();
break;
case 6:
traverse();
break;
case 7:
```

```
 search();
 break;
 case 8:
 exit(0);
 }
 }
return 0;
}
/*Function to create a new circular linked list*/
void create()
{
 int c;
x = (struct node*)malloc(sizeof(struct node));
 printf("\n Enter the data:");
 scanf("%d", &x->data);
 x->link = x;
 head = x;
 printf("\n If you wish to continue press 1 otherwise 0:");
 scanf("%d", &c);
 while (c != 0)
 {
 y = (struct node*)malloc(sizeof(struct node));
 printf("\n Enter the data:");
 scanf("%d", &y->data);
```

```
 x->link = y;
 y->link = head;
 x = y;
 printf("\n If you wish to continue press 1 otherwise 0:");
 scanf("%d", &c);
 }
}
/*Function to insert an element at the begining of the list*/
void ins_at_beg()
{
 x = head;
 y = (struct node*)malloc(sizeof(struct node));
 printf("\n Enter the data:");
 scanf("%d", &y->data);
 while (x->link != head)
 {
 x = x->link;
 }
 x->link = y;
 y->link = head;
 head = y;
}
/*Function to insert an element at any position the list*/
void ins_at_pos()
```

```
{
 struct node *ptr;
 int c = 1, pos, count = 1;
 y = (struct node*)malloc(sizeof(struct node));
 if (head == NULL)
 {
 printf("cannot enter an element at this place");
 }
 printf("\n Enter the data:");
 scanf("%d", &y->data);
 printf("\n Enter the position to be inserted:");
 scanf("%d", &pos);
 x = head;
 ptr = head;
 while (ptr->link != head)
 {
 count++;
 ptr = ptr->link;
 }
 count++;
 if (pos > count)
 {
 printf("OUT OF BOUND");
 return;
```

```
 }
 while (c < pos)
 {
 z = x;
 x = x->link;
 c++;
 }
 y->link = x;
 z->link = y;
}
/*Function to delete an element at any begining of the list*/
void del_at_beg()
{
 if (head == NULL)
 printf("\n List is empty");
 else
 {
 x = head;
 y = head;
 while (x->link != head)
 {
 x = x->link;
 }
 head = y->link;
```

```
 x->link = head;
 free(y);
 }
}
/*Function to delete an element at any position the list*/
void del_at_pos()
{
 if (head == NULL)
 printf("\n List is empty");
 else
 {
 int c = 1, pos;
 printf("\n Enter the position to be deleted:");
 scanf("%d", &pos);
 x = head;
 while (c < pos)
 {
 y = x;
 x = x->link;
 c++;
 }
 y->link = x->link;
 free(x);
 }
```

```
}
/*Function to display the elements in the list*/
void traverse()
{
 if (head == NULL)
 printf("\n List is empty");
 else
 {
 x = head;
 while (x->link != head)
 {
 printf("%d->", x->data);
 x = x->link;
 }
 printf("%d", x->data);
 }
}
/*Function to search an element in the list*/
void search()
{
 int search_val, count = 0, flag = 0;
 printf("\nenter the element to search\n");
 scanf("%d", &search_val);
 if (head == NULL)
```

```
printf("\nList is empty nothing to search");
else
{
x = head;
while (x->link != head)
{
if (x->data == search_val)
{
printf("\nthe element is found at %d", count);
flag = 1;
break;
}
count++;
x = x->link;
}
if (x->data == search_val)
{
printf("element found at postion %d", count);
}
if (flag == 0)
{
printf("\nelement not found");
}
}
```

}