PERSONAL CHECK LIST

SOURCE CODE:

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

typedef struct check check;
struct check {
char buffer[101];
check* next;
int count;
};
check* head = NULL;

int main()
{
    int i,j;
     for(i=0;i<3;i++)
      {
         for(j=1;j>0;j--)
      {

printf("\n\t\t\t<<<<<<<<<-------------:::::::::::----------------
>>>>>>>>>>>>>>>>\n");


printf("\n\t<<<<<<<<<<<<<<<---------::::::----\tWELCOME\t\t----:::::::--------
>>>>>>>>>>>>>>>>>\n\n\n");
printf("\t\t**\t**\t**\t**\t**\t**\t**\t**\t**\t**\n\n");
```

```
  }
}
int choice;
while(1) {
printf("1. View check list\n");
printf("2. Create new task\n");
printf("3. Finished one! Update list!\n");
printf("4. Exit");
printf("\n\n\nEnter your choice\t:\t");
scanf("%d", &choice);
switch (choice) {
case 1:
seetodo();
break;
case 2:
createtodo();
break;
case 3:
deletetodo();
break;
case 4:
exit(1);
break;
default:
printf("\nInvalid Choice :-(\n");
}
}
return 0;
}

seetodo()
{
check* temp;
temp = head;
if (head == NULL)
printf("\n\nAll Done! Enjoy your day! \n\n");
while (temp != NULL) {
printf("%d.)", temp->count);
puts(temp->buffer);
temp = temp->next;
}
printf("\n\n\n");
}
createtodo()
{
char c;
check *new, *temp;
while (1) {
printf("\nWant to add new task ?? If the system get stuck, Press any key to
continue or 'n' to return back to menu\n");
scanf("%c", &c);
if (c == 'n')
```

```
break;
else {
if (head == NULL) {
new = (check*)malloc(sizeof(check));
head = new;
printf("\nType chore.....\n");
gets(new->buffer);
new->count = 1;
head->next = NULL;
}
else {
temp = (check*)malloc(sizeof(check));
printf("\nType chore.....\n");
fflush(stdin);
gets(temp->buffer);
temp->next = NULL;

new->next = temp;
new = new->next;
}

adjustcount();
    }
  }
}

deletetodo()
```

```
{
int x;
check *del, *temp;
printf("\nEnter the chore number that you want to remove.\n\t\t");
if (head == NULL)
printf("\n\nThere is no chores to do for today :-)\n\n\n");
else {
scanf("%d", &x);
del = head;
temp = head->next;
while (1) {
if (del->count == x) {
head = head->next;
free(del);
adjustcount();
break;
}

if (temp->count == x) {
del->next = temp->next;
free(temp);
adjustcount();
break;
}
else {
del = temp;
temp = temp->next;
```

```
         }
       }
    }
}

adjustcount()
{
check* temp;
int i = 1;
temp = head;

    while (temp != NULL)
    {
        temp->count = i;
        i++;
        temp = temp->next;
    }
}
```