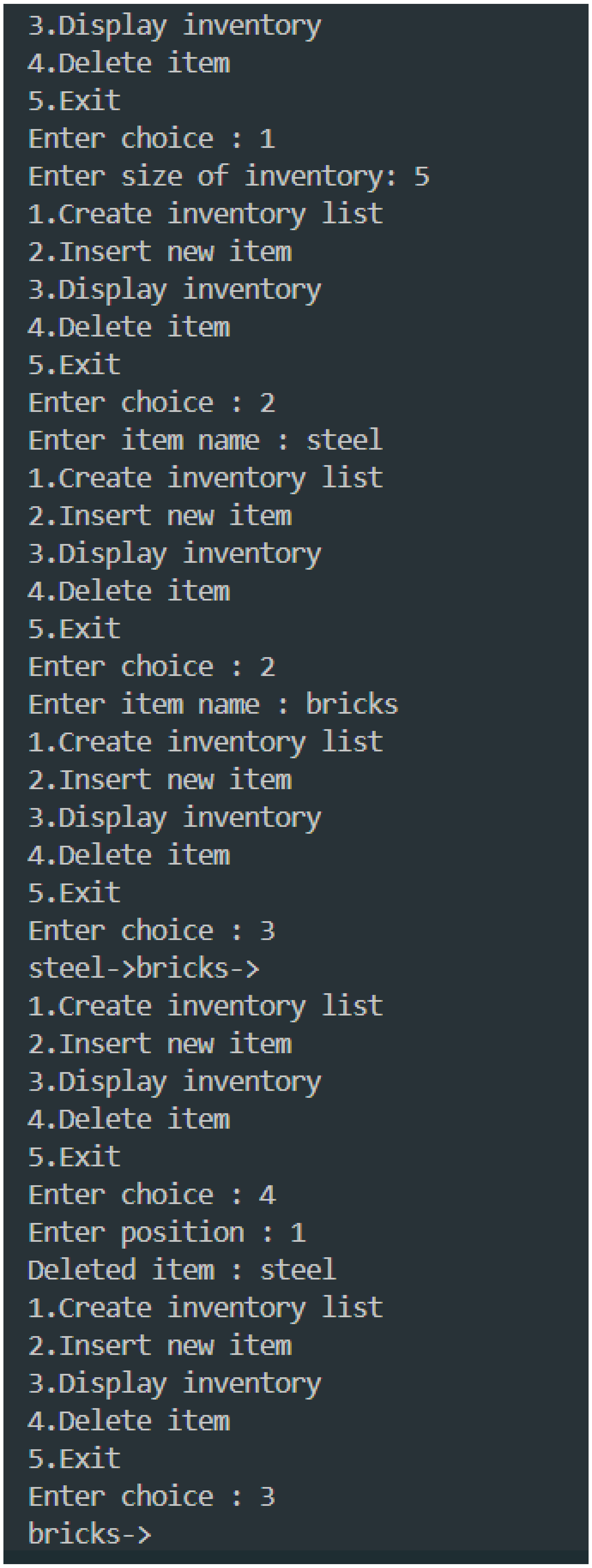
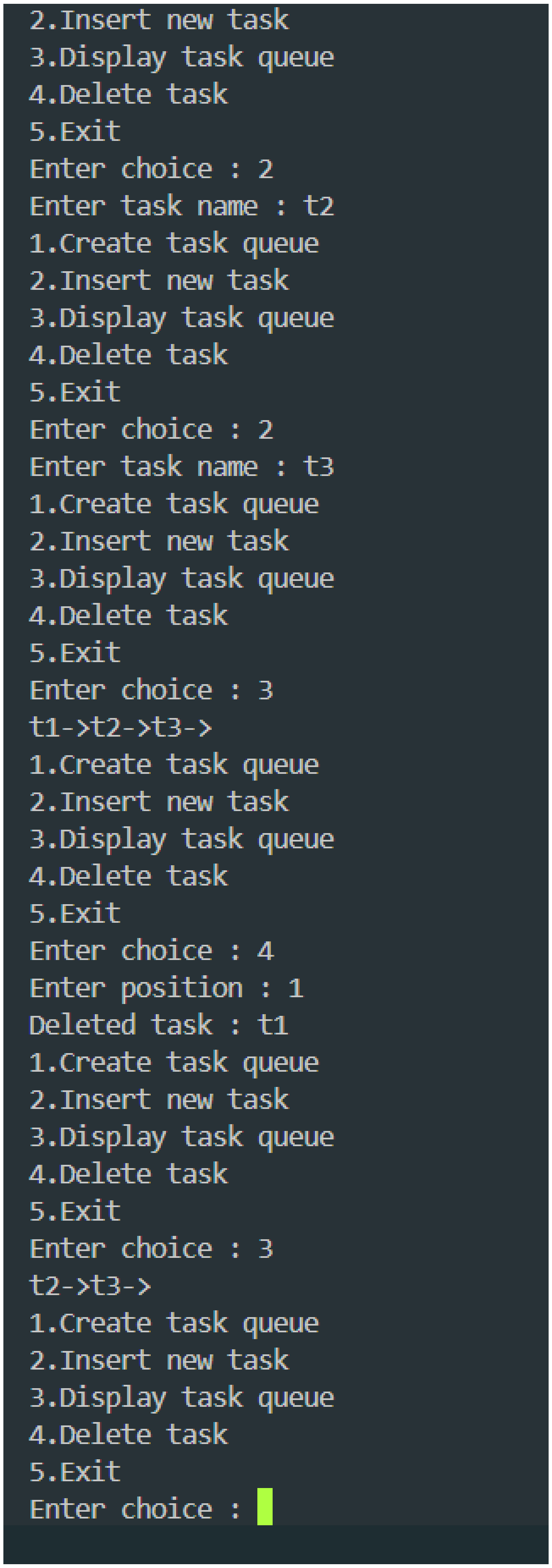
```
/*Implement a linked list to manage the inventory of raw materials.
Operations:
Create an inventory list.
Insert a new raw material.
Delete a raw material from the inventory.
Display the current inventory.*/
#include<stdlib.h>
#include<stdio.h>
#include<string.h>
struct Item
{
    char name[10];
    struct Item *next;
};
struct Item *first=NULL;
void create();
void insert();
void delete();
void display();
void main()
{
    int choice;
    do
    {
     printf("1.Create inventory list \n");
        printf("2.Insert new item\n");
        printf("3.Display inventory\n");
        printf("4.Delete item\n");
        printf("5.Exit\n");
        printf("Enter choice : ");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
            create();
            break;
            case 2:
            insert();
            break;
            case 3:
```

```c
            display();
            break;
            case 4:
            delete();
            break;
            case 5:printf("Exiing..\n");
            break;
        }
    } while (choice!=5);

}
void create ()
{
    int n;
    struct Item *ptr;
    ptr=first;
    printf("Enter size of inventory: ");
    scanf("%d",&n);
    for(int i=0;i<n;i++)
    {
        struct Item* newitem=(struct Item *)malloc(sizeof(struct Item));
        newitem->next=NULL;
        strcpy(newitem->name,"\0");
        if(first==NULL)
        {
            first=newitem;
            ptr=first;
        }
        else
        {
            ptr->next=newitem;
            ptr=ptr->next;
        }
    }
}
void insert()
{
    struct Item *ptr;
    char name[10];
    printf("Enter item name : ");
    scanf("%s",name);
    ptr=first;
    while(ptr!=0)
    {
        if(strcmp(ptr->name,"\0")==0)
        {
            strcpy(ptr->name,name);
            return;
        }
        ptr=ptr->next;
    }
}
void display()
{
    struct Item *ptr;
    ptr=first;
    while(ptr!=NULL)
    {
        if(strcmp(ptr->name,"\0")!=0)
        printf("%s->",ptr->name);
        ptr=ptr->next;
    }
    printf("\n");
}
void delete()
{
    int pos;
    struct Item*p,*q;
    p=first;
    printf("Enter position : ");
    scanf("%d",&pos);
    if(pos==1)
    {
        first=first->next;
        printf("Deleted item : %s\n",p->name);
        free(p);
    }
    else
    {
        for(int i=0;i<pos-1;i++)
```

```
     {
          q=p;
          p=p->next;
     }
     q->next=p->next;
     printf("Deleted item : %s\n",p->name);
     free(p);
     }


}
```


```
3.Display inventory
4.Delete item
5.Exit
Enter choice : 1
Enter size of inventory: 5
1.Create inventory list
2.Insert new item
3.Display inventory
4.Delete item
5.Exit
Enter choice : 2
Enter item name : steel
1.Create inventory list
2.Insert new item
3.Display inventory
4.Delete item
5.Exit
Enter choice : 2
Enter item name : bricks
1.Create inventory list
2.Insert new item
3.Display inventory
4.Delete item
5.Exit
Enter choice : 3
steel->bricks->
1.Create inventory list
2.Insert new item
3.Display inventory
4.Delete item
5.Exit
Enter choice : 4
Enter position : 1
Deleted item : steel
1.Create inventory list
2.Insert new item
3.Display inventory
4.Delete item
5.Exit
Enter choice : 3
bricks->
```

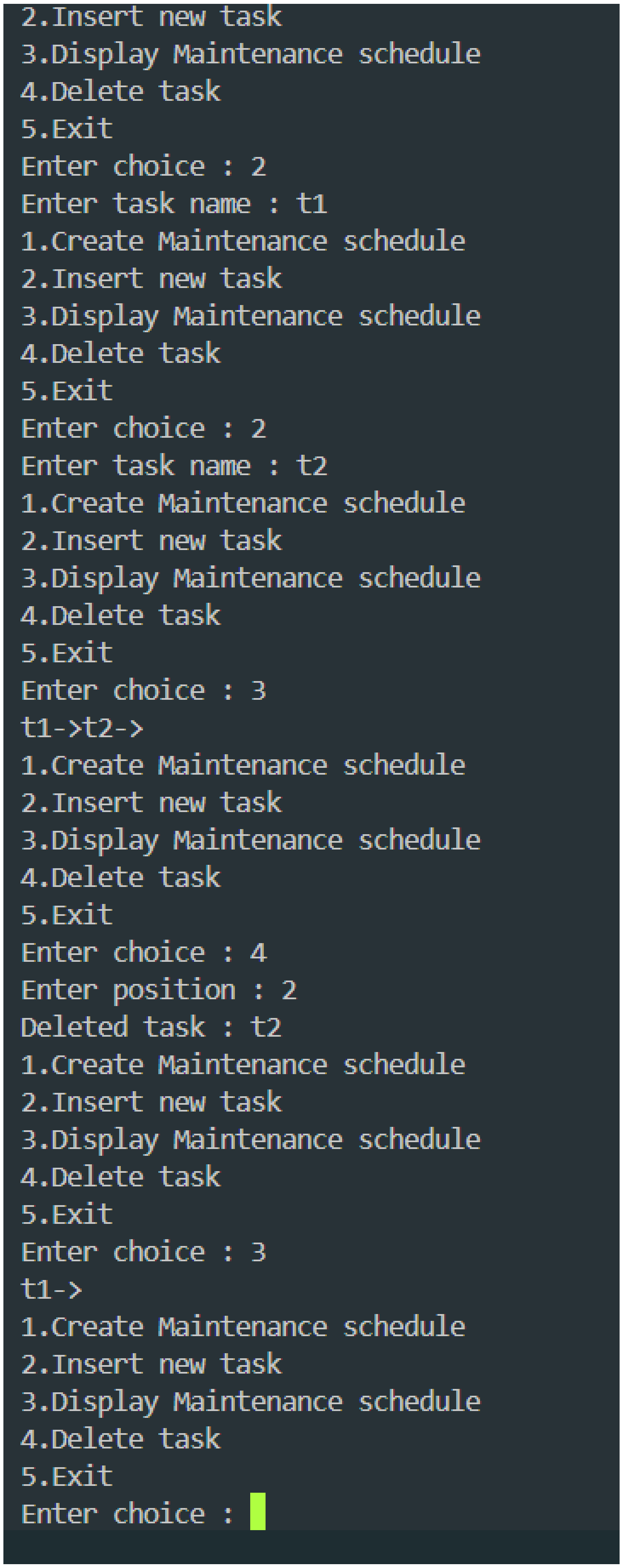
```c
/*Description: Use a linked list to manage the queue of tasks on a
production line.
Operations:
Create a production task queue.
Insert a new task into the queue.
Delete a completed task.
Display the current task queue.*/
#include<stdlib.h>
#include<stdio.h>
#include<string.h>
struct Task
{
    char name[20];
    struct Task *next;
};
struct Task *first=NULL;
void create();
void insert();
void delete();
void display();
void main()
{
    int choice;
    do
    {
     printf("1.Create task queue \n");
        printf("2.Insert new task\n");
        printf("3.Display task queue\n");
        printf("4.Delete task\n");
        printf("5.Exit\n");
        printf("Enter choice : ");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
            create();
            break;
            case 2:
            insert();
            break;
            case 3:
            display();
            break;
            case 4:
            delete();
            break;
            case 5:printf("Exiing..\n");
            break;
        }
    } while (choice!=5);

}
void create ()
{
    int n;
    struct Task *ptr;
    ptr=first;
    printf("Enter size of queue: ");
    scanf("%d",&n);
    for(int i=0;i<n;i++)
    {
        struct Task* newitem=(struct Task *)malloc(sizeof(struct Task));
        newitem->next=NULL;
        strcpy(newitem->name,"\0");
        if(first==NULL)
        {
            first=newitem;
            ptr=first;
        }
        else
        {
            ptr->next=newitem;
            ptr=ptr->next;
        }
    }
}
void insert()
{
    struct Task *ptr;
    char name[10];
    printf("Enter task name : ");
    scanf("%s",name);
    ptr=first;
    while(ptr!=0)
    {
        if(strcmp(ptr->name,"\0")==0)
        {
            strcpy(ptr->name,name);
            return;
        }
        ptr=ptr->next;
    }
}
void display()
{
    struct Task *ptr;
    ptr=first;
    while(ptr!=NULL)
    {
        if(strcmp(ptr->name,"\0")!=0)
        printf("%s->",ptr->name);
        ptr=ptr->next;
    }
    printf("\n");
}
void delete()
{
    int pos;
    struct Task*p,*q;
    p=first;
    printf("Enter position : ");
    scanf("%d",&pos);
    if(pos==1)
    {
        first=first->next;
        printf("Deleted task : %s\n",p->name);
        free(p);
    }
    else
    {
         for(int i=0;i<pos-1;i++)
    {
        q=p;
        p=p->next;
    }
    q->next=p->next;
    printf("Deleted task : %s\n",p->name);
    free(p);
    }

}
```



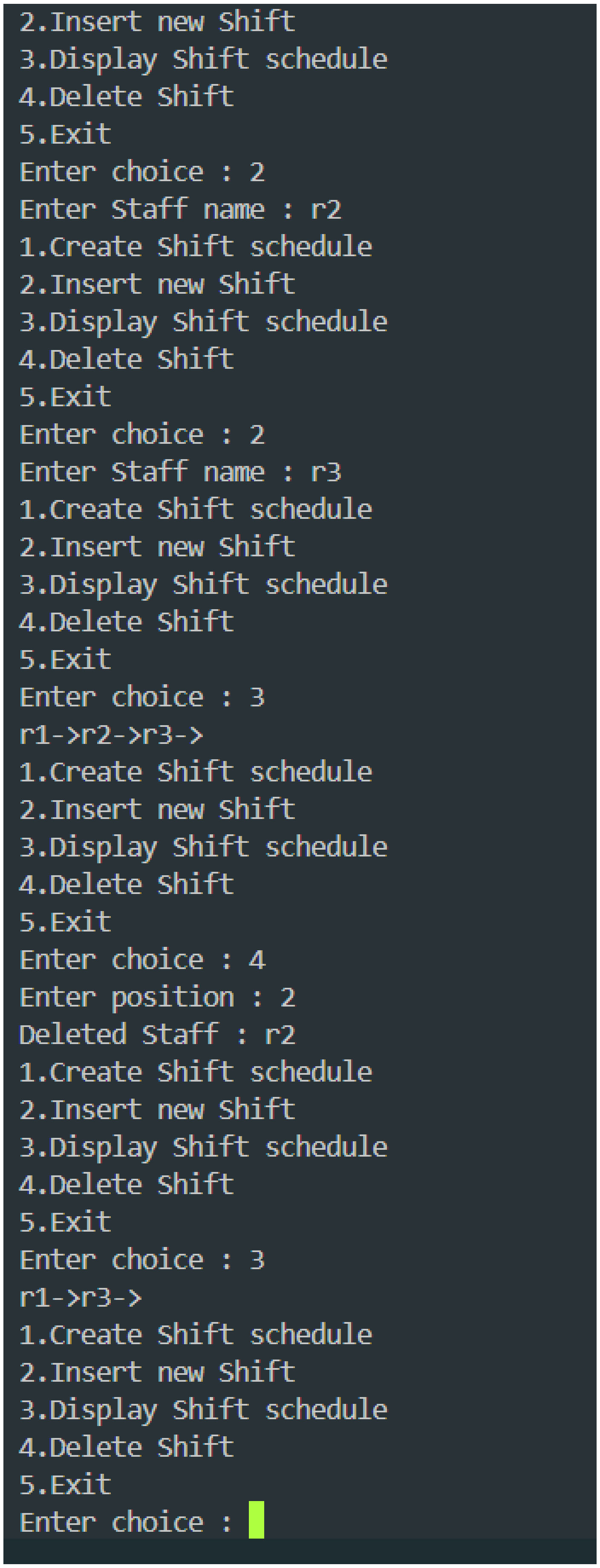
```
2.Insert new task
3.Display task queue
4.Delete task
5.Exit
Enter choice : 2
Enter task name : t2
1.Create task queue
2.Insert new task
3.Display task queue
4.Delete task
5.Exit
Enter choice : 2
Enter task name : t3
1.Create task queue
2.Insert new task
3.Display task queue
4.Delete task
5.Exit
Enter choice : 3
t1->t2->t3->
1.Create task queue
2.Insert new task
3.Display task queue
4.Delete task
5.Exit
Enter choice : 4
Enter position : 1
Deleted task : t1
1.Create task queue
2.Insert new task
3.Display task queue
4.Delete task
5.Exit
Enter choice : 3
t2->t3->
1.Create task queue
2.Insert new task
3.Display task queue
4.Delete task
5.Exit
Enter choice : 
```

```
/*Description: Develop a linked list to manage the maintenance schedule of
machines.
Operations:
Create a maintenance schedule.
Insert a new maintenance task.
Delete a completed maintenance task.
Display the maintenance schedule.*/
#include<stdlib.h>
#include<stdio.h>
#include<string.h>
struct Task
{
    char name[20];
    struct Task *next;
};
struct Task *first=NULL;
void create();
void insert();
void delete();
void display();
void main()
{
    int choice;
    do
    {
     printf("1.Create Maintenance schedule \n");
        printf("2.Insert new task\n");
        printf("3.Display Maintenance schedule\n");
        printf("4.Delete task\n");
        printf("5.Exit\n");
        printf("Enter choice : ");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
            create();
            break;
            case 2:
            insert();
            break;
            case 3:
            display();
            break;
            case 4:
            delete();
            break;
            case 5:printf("Exiing..\n");
            break;
        }
    } while (choice!=5);

}
void create ()
{
    int n;
    struct Task *ptr;
    ptr=first;
    printf("Enter size of schedule: ");
    scanf("%d",&n);
    for(int i=0;i<n;i++)
    {
        struct Task* newitem=(struct Task *)malloc(sizeof(struct Task));
        newitem->next=NULL;
        strcpy(newitem->name,"\0");
        if(first==NULL)
        {
            first=newitem;
            ptr=first;
        }
        else
        {
            ptr->next=newitem;
            ptr=ptr->next;
        }
    }
}
void insert()
{
    struct Task *ptr;
    char name[10];
```

```c
    printf("Enter task name : ");
    scanf("%s",name);
    ptr=first;
    while(ptr!=0)
    {
        if(strcmp(ptr->name,"\0")==0)
        {
            strcpy(ptr->name,name);
            return;
        }
        ptr=ptr->next;
    }
}
void display()
{
    struct Task *ptr;
    ptr=first;
    while(ptr!=NULL)
    {
        if(strcmp(ptr->name,"\0")!=0)
        printf("%s->",ptr->name);
        ptr=ptr->next;
    }
    printf("\n");
}
void delete()
{
    int pos;
    struct Task*p,*q;
    p=first;
    printf("Enter position : ");
    scanf("%d",&pos);
    if(pos==1)
    {
        first=first->next;
        printf("Deleted task : %s\n",p->name);
        free(p);
    }
    else
    {
         for(int i=0;i<pos-1;i++)
    {
        q=p;
        p=p->next;
    }
    q->next=p->next;
    printf("Deleted task : %s\n",p->name);
    free(p);
    }

}
```

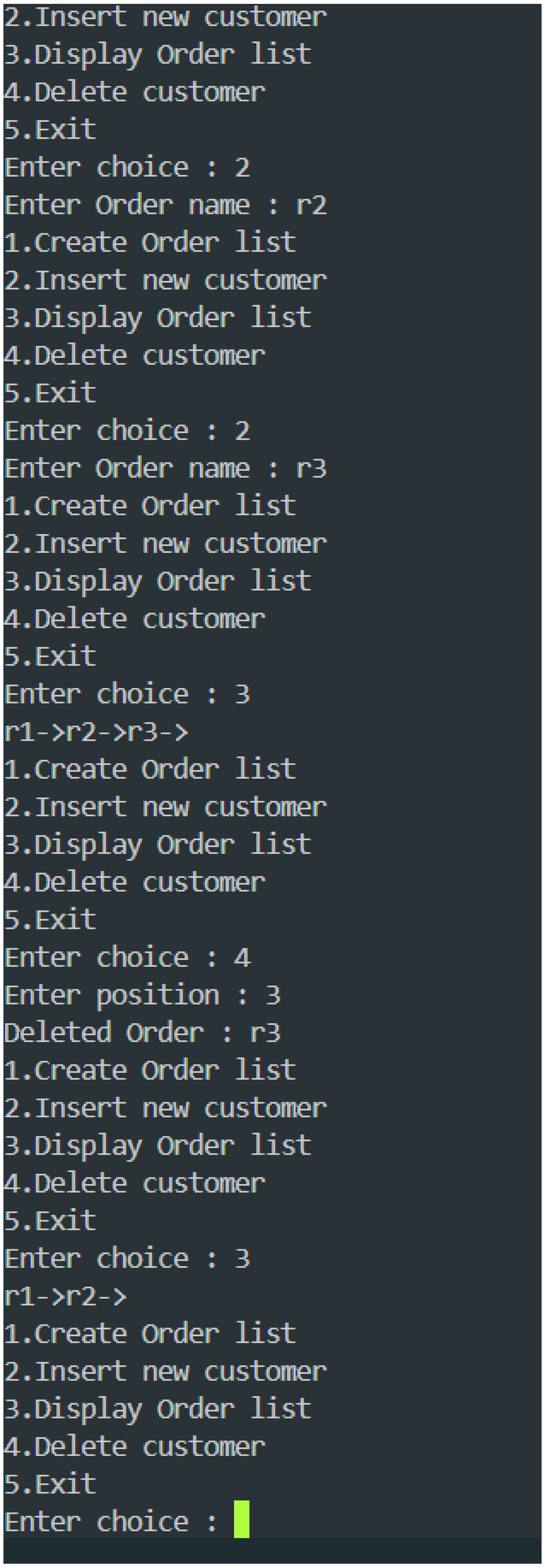

```
2.Insert new task
3.Display Maintenance schedule
4.Delete task
5.Exit
Enter choice : 2
Enter task name : t1
1.Create Maintenance schedule
2.Insert new task
3.Display Maintenance schedule
4.Delete task
5.Exit
Enter choice : 2
Enter task name : t2
1.Create Maintenance schedule
2.Insert new task
3.Display Maintenance schedule
4.Delete task
5.Exit
Enter choice : 3
t1->t2->
1.Create Maintenance schedule
2.Insert new task
3.Display Maintenance schedule
4.Delete task
5.Exit
Enter choice : 4
Enter position : 2
Deleted task : t2
1.Create Maintenance schedule
2.Insert new task
3.Display Maintenance schedule
4.Delete task
5.Exit
Enter choice : 3
t1->
1.Create Maintenance schedule
2.Insert new task
3.Display Maintenance schedule
4.Delete task
5.Exit
Enter choice :
```

```c
/*Description: Use a linked list to manage employee shifts in a
manufacturing plant.
Operations:
Create a shift schedule.
Insert a new shift.
Delete a completed or canceled shift.
Display the current shift schedule.*/
#include<stdlib.h>
#include<stdio.h>
#include<string.h>
struct Staff
{
    char name[20];
    struct Staff *next;
};
struct Staff *first=NULL;
void create();
void insert();
void delete();
void display();
void main()
{
    int choice;
    do
    {
     printf("1.Create Shift schedule \n");
        printf("2.Insert new Shift\n");
        printf("3.Display Shift schedule\n");
        printf("4.Delete Shift\n");
        printf("5.Exit\n");
        printf("Enter choice : ");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
            create();
            break;
            case 2:
            insert();
            break;
            case 3:
            display();
            break;
            case 4:
            delete();
            break;
            case 5:printf("Exiing..\n");
            break;
        }
    } while (choice!=5);

}
void create ()
{
    int n;
    struct Staff *ptr;
    ptr=first;
    printf("Enter size of schedule: ");
    scanf("%d",&n);
    for(int i=0;i<n;i++)
    {
        struct Staff* newitem=(struct Staff *)malloc(sizeof(struct
Staff));
        newitem->next=NULL;
        strcpy(newitem->name,"\0");
        if(first==NULL)
        {
            first=newitem;
            ptr=first;
        }
        else
        {
            ptr->next=newitem;
            ptr=ptr->next;
        }
    }
}
void insert()
{
    struct Staff *ptr;
    char name[10];
    printf("Enter Staff name : ");
    scanf("%s",name);
    ptr=first;
    while(ptr!=0)
    {
        if(strcmp(ptr->name,"\0")==0)
        {
            strcpy(ptr->name,name);
            return;
        }
        ptr=ptr->next;
    }
}
void display()
{
    struct Staff *ptr;
    ptr=first;
    while(ptr!=NULL)
    {
        if(strcmp(ptr->name,"\0")!=0)
        printf("%s->",ptr->name);
        ptr=ptr->next;
    }
    printf("\n");
}
void delete()
{
    int pos;
    struct Staff*p,*q;
    p=first;
    printf("Enter position : ");
    scanf("%d",&pos);
    if(pos==1)
    {
        first=first->next;
        printf("Deleted Staff : %s\n",p->name);
        free(p);
    }
    else
    {
         for(int i=0;i<pos-1;i++)
    {
        q=p;
        p=p->next;
    }
    q->next=p->next;
    printf("Deleted Staff : %s\n",p->name);
    free(p);
    }
}
```


```
2.Insert new Shift
3.Display Shift schedule
4.Delete Shift
5.Exit
Enter choice : 2
Enter Staff name : r2
1.Create Shift schedule
2.Insert new Shift
3.Display Shift schedule
4.Delete Shift
5.Exit
Enter choice : 2
Enter Staff name : r3
1.Create Shift schedule
2.Insert new Shift
3.Display Shift schedule
4.Delete Shift
5.Exit
Enter choice : 3
r1->r2->r3->
1.Create Shift schedule
2.Insert new Shift
3.Display Shift schedule
4.Delete Shift
5.Exit
Enter choice : 4
Enter position : 2
Deleted Staff : r2
1.Create Shift schedule
2.Insert new Shift
3.Display Shift schedule
4.Delete Shift
5.Exit
Enter choice : 3
r1->r3->
1.Create Shift schedule
2.Insert new Shift
3.Display Shift schedule
4.Delete Shift
5.Exit
Enter choice :
```

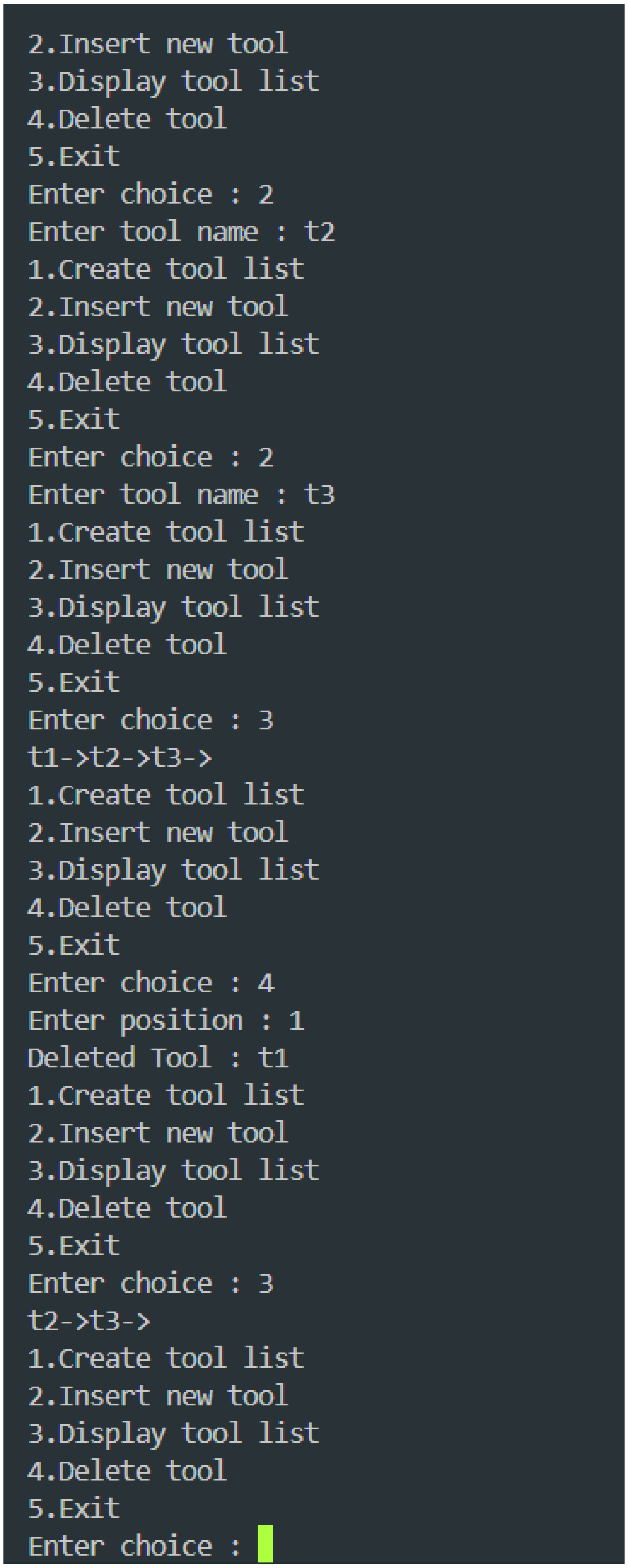
```
/*Description: Implement a linked list to track customer orders.
Operations:
Create an order list.
Insert a new customer order.
Delete a completed or canceled order.
Display all current orders*/
#include<stdlib.h>
#include<stdio.h>
#include<string.h>
struct Order
{
    char name[20];
    struct Order *next;
};
struct Order *first=NULL;
void create();
void insert();
void delete();
void display();
void main()
{
    int choice;
    do
    {
     printf("1.Create Order list \n");
        printf("2.Insert new customer\n");
        printf("3.Display Order list\n");
        printf("4.Delete customer\n");
        printf("5.Exit\n");
        printf("Enter choice : ");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
            create();
            break;
            case 2:
            insert();
            break;
            case 3:
```

```
            display();
            break;
            case 4:
            delete();
            break;
            case 5:printf("Exiing..\n");
            break;
        }
    } while (choice!=5);

}
void create ()
{
    int n;
    struct Order *ptr;
    ptr=first;
    printf("Enter size of order list: ");
    scanf("%d",&n);
    for(int i=0;i<n;i++)
    {
        struct Order* newitem=(struct Order *)malloc(sizeof(struct
Order));
        newitem->next=NULL;
        strcpy(newitem->name,"\0");
        if(first==NULL)
        {
            first=newitem;
            ptr=first;
        }
        else
        {
            ptr->next=newitem;
            ptr=ptr->next;
        }
    }
}
void insert()
{
    struct Order *ptr;
    char name[10];
    printf("Enter Order name : ");
    scanf("%s",name);
    ptr=first;
    while(ptr!=0)
    {
        if(strcmp(ptr->name,"\0")==0)
        {
            strcpy(ptr->name,name);
            return;
        }
        ptr=ptr->next;
    }
}
void display()
{
    struct Order *ptr;
    ptr=first;
    while(ptr!=NULL)
    {
        if(strcmp(ptr->name,"\0")!=0)
        printf("%s->",ptr->name);
        ptr=ptr->next;
    }
    printf("\n");
}
void delete()
{
    int pos;
    struct Order*p,*q;
    p=first;
    printf("Enter position : ");
    scanf("%d",&pos);
    if(pos==1)
    {
        first=first->next;
        printf("Deleted Order : %s\n",p->name);
        free(p);
    }
    else
    {
         for(int i=0;i<pos-1;i++)
    {
        q=p;
        p=p->next;
    }
    q->next=p->next;
    printf("Deleted Order : %s\n",p->name);
    free(p);
    }
}
```



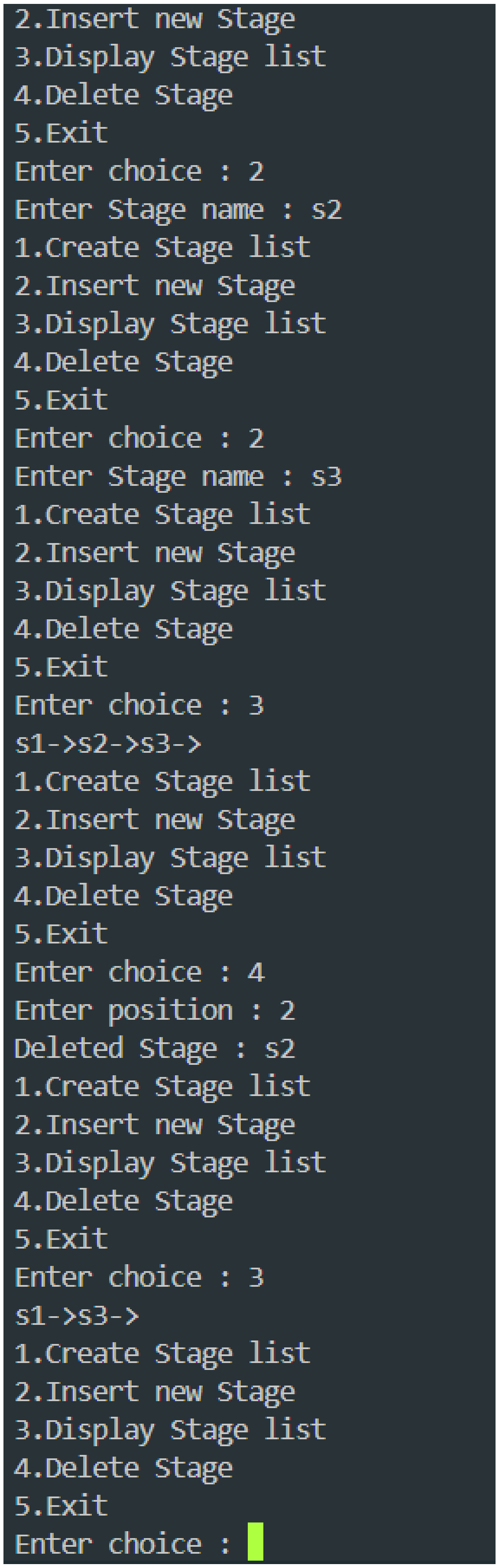
```
2.Insert new customer
3.Display Order list
4.Delete customer
5.Exit
Enter choice : 2
Enter Order name : r2
1.Create Order list
2.Insert new customer
3.Display Order list
4.Delete customer
5.Exit
Enter choice : 2
Enter Order name : r3
1.Create Order list
2.Insert new customer
3.Display Order list
4.Delete customer
5.Exit
Enter choice : 3
r1->r2->r3->
1.Create Order list
2.Insert new customer
3.Display Order list
4.Delete customer
5.Exit
Enter choice : 4
Enter position : 3
Deleted Order : r3
1.Create Order list
2.Insert new customer
3.Display Order list
4.Delete customer
5.Exit
Enter choice : 3
r1->r2->
1.Create Order list
2.Insert new customer
3.Display Order list
4.Delete customer
5.Exit
Enter choice :
```

```
/*Description: Maintain a linked list to track tools used in the
manufacturing process.
Operations:
Create a tool tracking list.
Insert a new tool entry.
Delete a tool that is no longer in use.
Display all tools currently tracked.*/
#include<stdlib.h>
#include<stdio.h>
#include<string.h>
struct Tool
{
    char name[20];
    struct Tool *next;
};
struct Tool *first=NULL;
void create();
void insert();
void delete();
void display();
void main()
{
    int choice;
    do
    {
     printf("1.Create tool list \n");
        printf("2.Insert new tool\n");
        printf("3.Display tool list\n");
        printf("4.Delete tool\n");
        printf("5.Exit\n");
        printf("Enter choice : ");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
            create();
            break;
            case 2:
            insert();
            break;
            case 3:
            display();
            break;
            case 4:
            delete();
            break;
            case 5:printf("Exiing..\n");
            break;
        }
    } while (choice!=5);

}
void create ()
{
    int n;
    struct Tool *ptr;
    ptr=first;
    printf("Enter size of tool list: ");
    scanf("%d",&n);
    for(int i=0;i<n;i++)
    {
        struct Tool* newitem=(struct Tool *)malloc(sizeof(struct Tool));
        newitem->next=NULL;
        strcpy(newitem->name,"\0");
        if(first==NULL)
        {
            first=newitem;
            ptr=first;
        }
        else
        {
            ptr->next=newitem;
            ptr=ptr->next;
        }
    }
}
void insert()
{
    struct Tool *ptr;
    char name[10];
    printf("Enter tool name : ");
    scanf("%s",name);
    ptr=first;
    while(ptr!=0)
    {
        if(strcmp(ptr->name,"\0")==0)
        {
            strcpy(ptr->name,name);
            return;
        }
        ptr=ptr->next;
    }
}
void display()
{
    struct Tool *ptr;
    ptr=first;
    while(ptr!=NULL)
    {
        if(strcmp(ptr->name,"\0")!=0)
        printf("%s->",ptr->name);
        ptr=ptr->next;
    }
    printf("\n");
}
void delete()
{
    int pos;
    struct Tool*p,*q;
    p=first;
    printf("Enter position : ");
    scanf("%d",&pos);
    if(pos==1)
    {
        first=first->next;
        printf("Deleted Tool : %s\n",p->name);
        free(p);
    }
    else
    {
         for(int i=0;i<pos-1;i++)
    {
        q=p;
        p=p->next;
    }
    q->next=p->next;
    printf("Deleted Tool : %s\n",p->name);
    free(p);
    }
}
```



```
2.Insert new tool
3.Display tool list
4.Delete tool
5.Exit
Enter choice : 2
Enter tool name : t2
1.Create tool list
2.Insert new tool
3.Display tool list
4.Delete tool
5.Exit
Enter choice : 2
Enter tool name : t3
1.Create tool list
2.Insert new tool
3.Display tool list
4.Delete tool
5.Exit
Enter choice : 3
t1->t2->t3->
1.Create tool list
2.Insert new tool
3.Display tool list
4.Delete tool
5.Exit
Enter choice : 4
Enter position : 1
Deleted Tool : t1
1.Create tool list
2.Insert new tool
3.Display tool list
4.Delete tool
5.Exit
Enter choice : 3
t2->t3->
1.Create tool list
2.Insert new tool
3.Display tool list
4.Delete tool
5.Exit
Enter choice : 
```

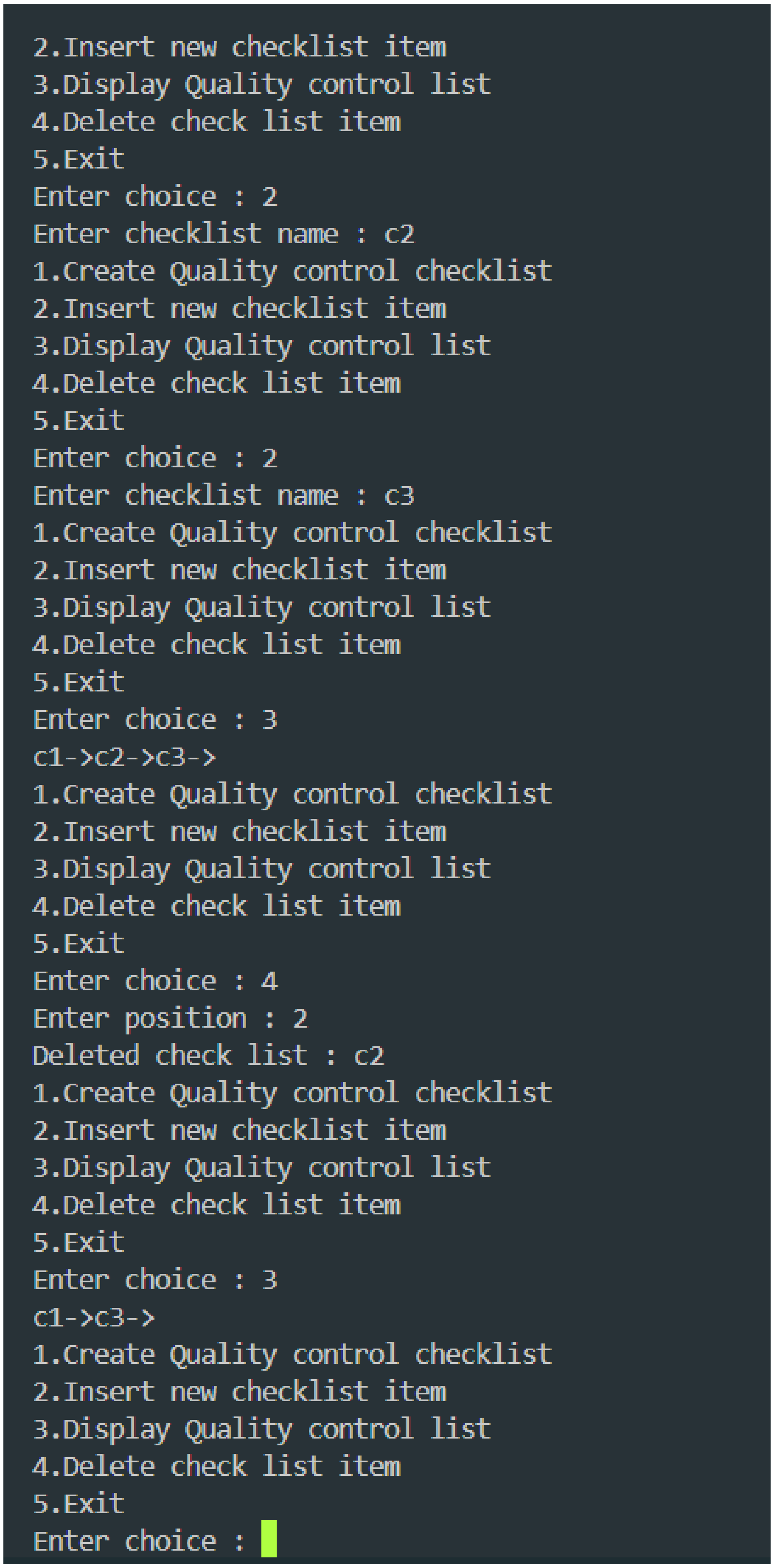
```
/*Description: Use a linked list to manage the assembly stages of a
product.
Operations:
Create an assembly line stage list.
Insert a new stage.
Delete a completed stage.
Display the current assembly stages*/
#include<stdlib.h>
#include<stdio.h>
#include<string.h>
struct Stage
{
    char name[20];
    struct Stage *next;
};
struct Stage *first=NULL;
void create();
void insert();
void delete();
void display();
void main()
{
    int choice;
    do
    {
     printf("1.Create Stage list \n");
        printf("2.Insert new Stage\n");
        printf("3.Display Stage list\n");
        printf("4.Delete Stage\n");
        printf("5.Exit\n");
        printf("Enter choice : ");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
            create();
            break;
            case 2:
            insert();
            break;
            case 3:
            display();
            break;
            case 4:
            delete();
            break;
            case 5:printf("Exiing..\n");
            break;
        }
    } while (choice!=5);

}
void create ()
{
    int n;
    struct Stage *ptr;
    ptr=first;
    printf("Enter size of Stage list: ");
    scanf("%d",&n);
    for(int i=0;i<n;i++)
    {
        struct Stage* newitem=(struct Stage *)malloc(sizeof(struct
Stage));
        newitem->next=NULL;
        strcpy(newitem->name,"\0");
        if(first==NULL)
        {
            first=newitem;
            ptr=first;
        }
        else
        {
            ptr->next=newitem;
            ptr=ptr->next;
        }
    }
}
void insert()
{
    struct Stage *ptr;
    char name[10];
    printf("Enter Stage name : ");
    scanf("%s",name);
    ptr=first;
    while(ptr!=0)
    {
        if(strcmp(ptr->name,"\0")==0)
        {
            strcpy(ptr->name,name);
            return;
        }
        ptr=ptr->next;
    }
}
void display()
{
    struct Stage *ptr;
    ptr=first;
    while(ptr!=NULL)
    {
        if(strcmp(ptr->name,"\0")!=0)
        printf("%s->",ptr->name);
        ptr=ptr->next;
    }
    printf("\n");
}
void delete()
{
    int pos;
    struct Stage*p,*q;
    p=first;
    printf("Enter position : ");
    scanf("%d",&pos);
    if(pos==1)
    {
        first=first->next;
        printf("Deleted Stage : %s\n",p->name);
        free(p);
    }
    else
    {
         for(int i=0;i<pos-1;i++)
    {
        q=p;
        p=p->next;
    }
    q->next=p->next;
    printf("Deleted Stage : %s\n",p->name);
    free(p);
    }
}
```


```
2.Insert new Stage
3.Display Stage list
4.Delete Stage
5.Exit
Enter choice : 2
Enter Stage name : s2
1.Create Stage list
2.Insert new Stage
3.Display Stage list
4.Delete Stage
5.Exit
Enter choice : 2
Enter Stage name : s3
1.Create Stage list
2.Insert new Stage
3.Display Stage list
4.Delete Stage
5.Exit
Enter choice : 3
s1->s2->s3->
1.Create Stage list
2.Insert new Stage
3.Display Stage list
4.Delete Stage
5.Exit
Enter choice : 4
Enter position : 2
Deleted Stage : s2
1.Create Stage list
2.Insert new Stage
3.Display Stage list
4.Delete Stage
5.Exit
Enter choice : 3
s1->s3->
1.Create Stage list
2.Insert new Stage
3.Display Stage list
4.Delete Stage
5.Exit
Enter choice :
```

```
/*Implement a linked list to manage a quality control checklist.
Operations:
Create a quality control checklist.
Insert a new checklist item.
Delete a completed or outdated checklist item.
Display the current quality control checklist*/
#include<stdlib.h>
#include<stdio.h>
#include<string.h>
struct Quality
{
    char name[20];
    struct Quality *next;
};
struct Quality *first=NULL;
void create();
void insert();
void delete();
void display();
void main()
{
    int choice;
    do
    {
     printf("1.Create Quality control checklist \n");
        printf("2.Insert new checklist item\n");
        printf("3.Display Quality control list\n");
        printf("4.Delete check list item\n");
        printf("5.Exit\n");
        printf("Enter choice : ");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
            create();
            break;
            case 2:
            insert();
            break;
            case 3:
```

```
            display();
            break;
            case 4:
            delete();
            break;
            case 5:printf("Exiing..\n");
            break;
        }
    } while (choice!=5);

}
void create ()
{
    int n;
    struct Quality *ptr;
    ptr=first;
    printf("Enter size of Quality control list: ");
    scanf("%d",&n);
    for(int i=0;i<n;i++)
    {
        struct Quality* newitem=(struct Quality *)malloc(sizeof(struct
Quality));
        newitem->next=NULL;
        strcpy(newitem->name,"\0");
        if(first==NULL)
        {
            first=newitem;
            ptr=first;
        }
        else
        {
            ptr->next=newitem;
            ptr=ptr->next;
        }
    }
}
void insert()
{
    struct Quality *ptr;
    char name[10];
```

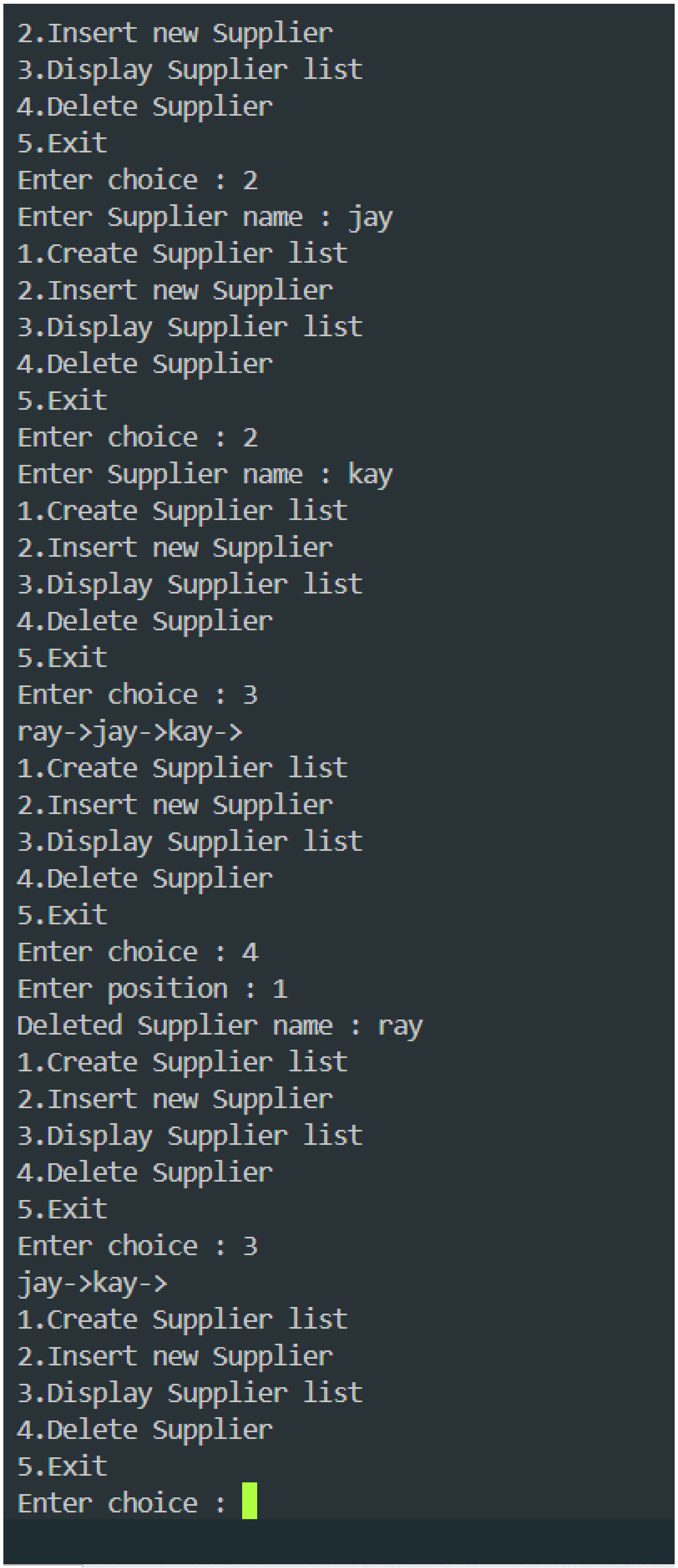
```c
    printf("Enter checklist name : ");
    scanf("%s",name);
    ptr=first;
    while(ptr!=0)
    {
        if(strcmp(ptr->name,"\0")==0)
        {
            strcpy(ptr->name,name);
            return;
        }
        ptr=ptr->next;
    }
}
void display()
{
    struct Quality *ptr;
    ptr=first;
    while(ptr!=NULL)
    {
        if(strcmp(ptr->name,"\0")!=0)
        printf("%s->",ptr->name);
        ptr=ptr->next;
    }
    printf("\n");
}
void delete()
{
    int pos;
    struct Quality*p,*q;
    p=first;
    printf("Enter position : ");
    scanf("%d",&pos);
    if(pos==1)
    {
        first=first->next;
        printf("Deleted check list : %s\n",p->name);
        free(p);
    }
    else
    {
          for(int i=0;i<pos-1;i++)
    {
        q=p;
        p=p->next;
    }
    q->next=p->next;
    printf("Deleted check list : %s\n",p->name);
    free(p);
    }
}
```


```
2.Insert new checklist item
3.Display Quality control list
4.Delete check list item
5.Exit
Enter choice : 2
Enter checklist name : c2
1.Create Quality control checklist
2.Insert new checklist item
3.Display Quality control list
4.Delete check list item
5.Exit
Enter choice : 2
Enter checklist name : c3
1.Create Quality control checklist
2.Insert new checklist item
3.Display Quality control list
4.Delete check list item
5.Exit
Enter choice : 3
c1->c2->c3->
1.Create Quality control checklist
2.Insert new checklist item
3.Display Quality control list
4.Delete check list item
5.Exit
Enter choice : 4
Enter position : 2
Deleted check list : c2
1.Create Quality control checklist
2.Insert new checklist item
3.Display Quality control list
4.Delete check list item
5.Exit
Enter choice : 3
c1->c3->
1.Create Quality control checklist
2.Insert new checklist item
3.Display Quality control list
4.Delete check list item
5.Exit
Enter choice : 
```

```c
/*Use a linked list to manage a list of suppliers.
Operations:
Create a supplier list.
Insert a new supplier.
Delete an inactive or outdated supplier.
Display all current suppliers.*/
#include<stdlib.h>
#include<stdio.h>
#include<string.h>
struct Supplier
{
    char name[20];
    struct Supplier *next;
};
struct Supplier *first=NULL;
void create();
void insert();
void delete();
void display();
void main()
{
    int choice;
    do
    {
     printf("1.Create Supplier list \n");
        printf("2.Insert new Supplier\n");
        printf("3.Display Supplier list\n");
        printf("4.Delete Supplier\n");
        printf("5.Exit\n");
        printf("Enter choice : ");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
            create();
            break;
            case 2:
            insert();
            break;
            case 3:
```

```
            display();
            break;
            case 4:
            delete();
            break;
            case 5:printf("Exiing..\n");
            break;
        }
    } while (choice!=5);

}
void create ()
{
    int n;
    struct Supplier *ptr;
    ptr=first;
    printf("Enter size of Supplier list: ");
    scanf("%d",&n);
    for(int i=0;i<n;i++)
    {
        struct Supplier* newitem=(struct Supplier *)malloc(sizeof(struct
Supplier));
        newitem->next=NULL;
        strcpy(newitem->name,"\0");
        if(first==NULL)
        {
            first=newitem;
            ptr=first;
        }
        else
        {
            ptr->next=newitem;
            ptr=ptr->next;
        }
    }
}
void insert()
{
    struct Supplier *ptr;
    char name[10];
    printf("Enter Supplier name : ");
    scanf("%s",name);
    ptr=first;
    while(ptr!=0)
    {
        if(strcmp(ptr->name,"\0")==0)
        {
            strcpy(ptr->name,name);
            return;
        }
        ptr=ptr->next;
    }
}
void display()
{
    struct Supplier *ptr;
    ptr=first;
    while(ptr!=NULL)
    {
        if(strcmp(ptr->name,"\0")!=0)
        printf("%s->",ptr->name);
        ptr=ptr->next;
    }
    printf("\n");
}
void delete()
{
    int pos;
    struct Supplier*p,*q;
    p=first;
    printf("Enter position : ");
    scanf("%d",&pos);
    if(pos==1)
    {
        first=first->next;
        printf("Deleted Supplier name : %s\n",p->name);
        free(p);
    }
    else
    {
```

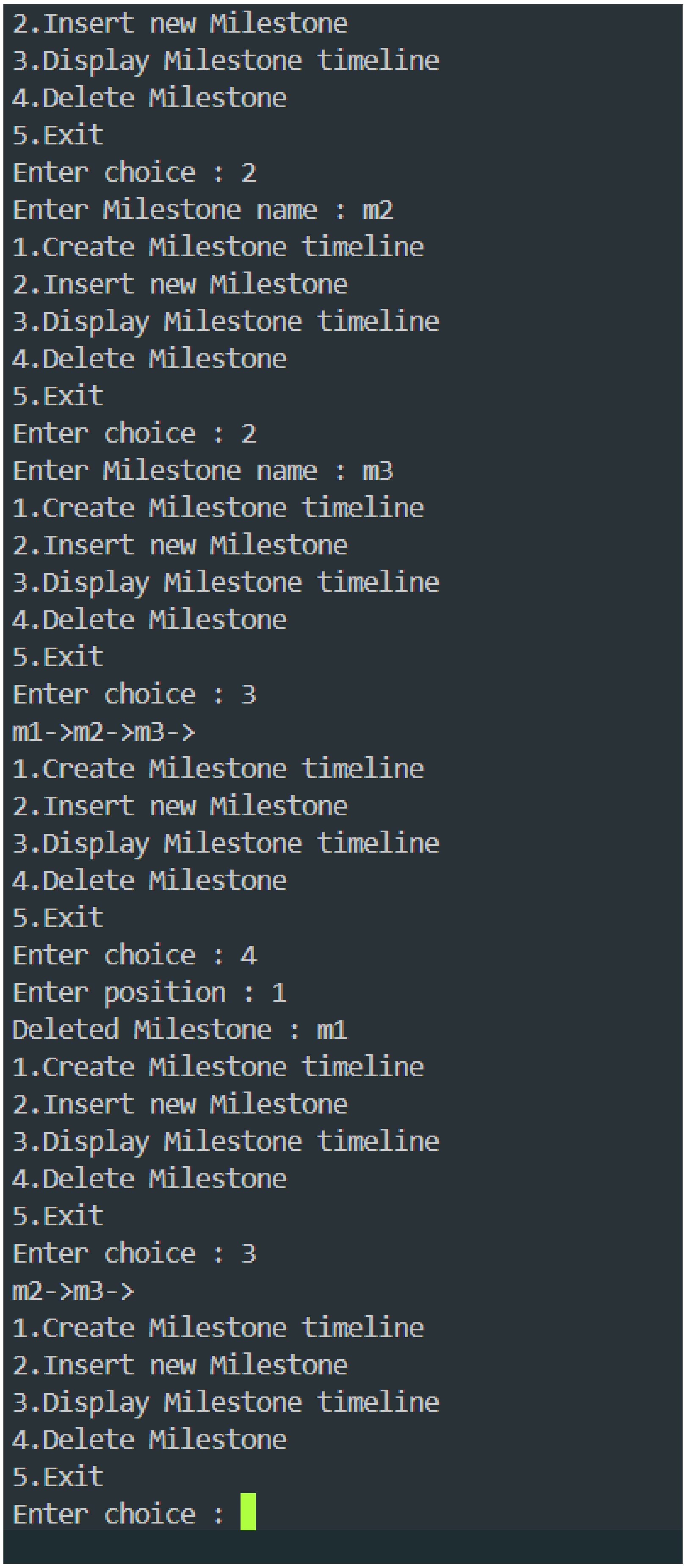
```c
         for(int i=0;i<pos-1;i++)
    {
        q=p;
        p=p->next;
    }
    q->next=p->next;
    printf("Deleted Supplier name : %s\n",p->name);
    free(p);
    }
}
```



```
2.Insert new Supplier
3.Display Supplier list
4.Delete Supplier
5.Exit
Enter choice : 2
Enter Supplier name : jay
1.Create Supplier list
2.Insert new Supplier
3.Display Supplier list
4.Delete Supplier
5.Exit
Enter choice : 2
Enter Supplier name : kay
1.Create Supplier list
2.Insert new Supplier
3.Display Supplier list
4.Delete Supplier
5.Exit
Enter choice : 3
ray->jay->kay->
1.Create Supplier list
2.Insert new Supplier
3.Display Supplier list
4.Delete Supplier
5.Exit
Enter choice : 4
Enter position : 1
Deleted Supplier name : ray
1.Create Supplier list
2.Insert new Supplier
3.Display Supplier list
4.Delete Supplier
5.Exit
Enter choice : 3
jay->kay->
1.Create Supplier list
2.Insert new Supplier
3.Display Supplier list
4.Delete Supplier
5.Exit
Enter choice : 
```

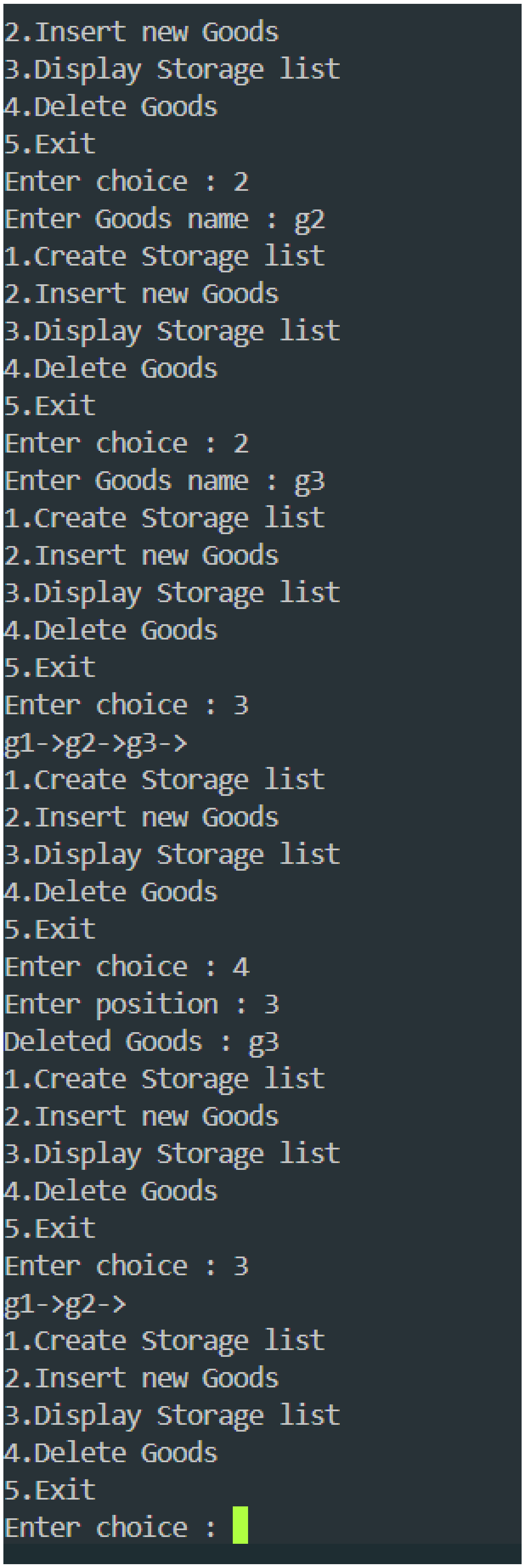
```c
/*Develop a linked list to manage the timeline of a manufacturing project.
Operations:
Create a project timeline.
Insert a new project milestone.
Delete a completed milestone.
Display the current project timeline.*/
#include<stdlib.h>
#include<stdio.h>
#include<string.h>
struct Milestone
{
    char name[20];
    struct Milestone *next;
};
struct Milestone *first=NULL;
void create();
void insert();
void delete();
void display();
void main()
{
    int choice;
    do
    {
     printf("1.Create Milestone timeline \n");
        printf("2.Insert new Milestone\n");
        printf("3.Display Milestone timeline\n");
        printf("4.Delete Milestone\n");
        printf("5.Exit\n");
        printf("Enter choice : ");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
            create();
            break;
            case 2:
            insert();
            break;
            case 3:
```

```
            display();
            break;
            case 4:
            delete();
            break;
            case 5:printf("Exiing..\n");
            break;
        }
    } while (choice!=5);

}
void create ()
{
    int n;
    struct Milestone *ptr;
    ptr=first;
    printf("Enter size of Milestone timeline: ");
    scanf("%d",&n);
    for(int i=0;i<n;i++)
    {
        struct Milestone* newitem=(struct Milestone *)malloc(sizeof(struct
Milestone));
        newitem->next=NULL;
        strcpy(newitem->name,"\0");
        if(first==NULL)
        {
            first=newitem;
            ptr=first;
        }
        else
        {
            ptr->next=newitem;
            ptr=ptr->next;
        }
    }
}
void insert()
{
    struct Milestone *ptr;
    char name[10];
    printf("Enter Milestone name : ");
    scanf("%s",name);
    ptr=first;
    while(ptr!=0)
    {
        if(strcmp(ptr->name,"\0")==0)
        {
            strcpy(ptr->name,name);
            return;
        }
        ptr=ptr->next;
    }
}
void display()
{
    struct Milestone *ptr;
    ptr=first;
    while(ptr!=NULL)
    {
        if(strcmp(ptr->name,"\0")!=0)
        printf("%s->",ptr->name);
        ptr=ptr->next;
    }
    printf("\n");
}
void delete()
{
    int pos;
    struct Milestone*p,*q;
    p=first;
    printf("Enter position : ");
    scanf("%d",&pos);
    if(pos==1)
    {
        first=first->next;
        printf("Deleted Milestone : %s\n",p->name);
        free(p);
    }
    else
    {
          for(int i=0;i<pos-1;i++)
    {
        q=p;
        p=p->next;
    }
    q->next=p->next;
    printf("Deleted Milestone : %s\n",p->name);
    free(p);
    }
}
```


```
2.Insert new Milestone
3.Display Milestone timeline
4.Delete Milestone
5.Exit
Enter choice : 2
Enter Milestone name : m2
1.Create Milestone timeline
2.Insert new Milestone
3.Display Milestone timeline
4.Delete Milestone
5.Exit
Enter choice : 2
Enter Milestone name : m3
1.Create Milestone timeline
2.Insert new Milestone
3.Display Milestone timeline
4.Delete Milestone
5.Exit
Enter choice : 3
m1->m2->m3->
1.Create Milestone timeline
2.Insert new Milestone
3.Display Milestone timeline
4.Delete Milestone
5.Exit
Enter choice : 4
Enter position : 1
Deleted Milestone : m1
1.Create Milestone timeline
2.Insert new Milestone
3.Display Milestone timeline
4.Delete Milestone
5.Exit
Enter choice : 3
m2->m3->
1.Create Milestone timeline
2.Insert new Milestone
3.Display Milestone timeline
4.Delete Milestone
5.Exit
Enter choice :
```

```
/*Implement a linked list to manage the storage of goods in a warehouse.
Operations:
Create a storage list.
Insert a new storage entry.
Delete a storage entry when goods are shipped.
Display the current warehouse storage.*/
#include<stdlib.h>
#include<stdio.h>
#include<string.h>
struct Goods
{
    char name[20];
    struct Goods *next;
};
struct Goods *first=NULL;
void create();
void insert();
void delete();
void display();
void main()
{
    int choice;
    do
    {
     printf("1.Create Storage list \n");
        printf("2.Insert new Goods\n");
        printf("3.Display Storage list\n");
        printf("4.Delete Goods\n");
        printf("5.Exit\n");
        printf("Enter choice : ");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
            create();
            break;
            case 2:
            insert();
            break;
            case 3:
```

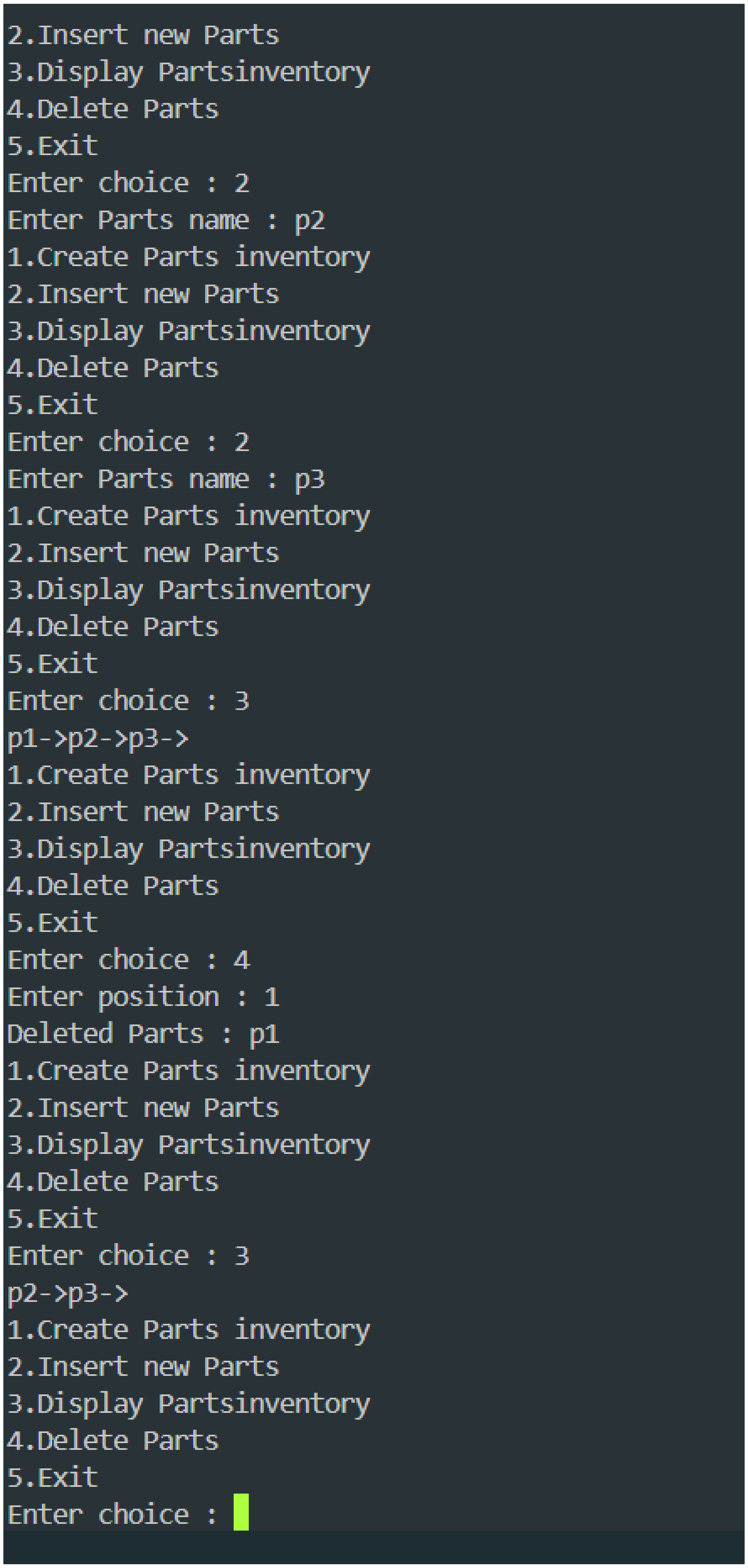
```
            display();
            break;
            case 4:
            delete();
            break;
            case 5:printf("Exiing..\n");
            break;
        }
    } while (choice!=5);

}
void create ()
{
    int n;
    struct Goods *ptr;
    ptr=first;
    printf("Enter size of Storage list: ");
    scanf("%d",&n);
    for(int i=0;i<n;i++)
    {
        struct Goods* newitem=(struct Goods *)malloc(sizeof(struct
Goods));
        newitem->next=NULL;
        strcpy(newitem->name,"\0");
        if(first==NULL)
        {
            first=newitem;
            ptr=first;
        }
        else
        {
            ptr->next=newitem;
            ptr=ptr->next;
        }
    }
}
void insert()
{
    struct Goods *ptr;
    char name[10];
    printf("Enter Goods name : ");
    scanf("%s",name);
    ptr=first;
    while(ptr!=0)
    {
        if(strcmp(ptr->name,"\0")==0)
        {
            strcpy(ptr->name,name);
            return;
        }
        ptr=ptr->next;
    }
}
void display()
{
    struct Goods *ptr;
    ptr=first;
    while(ptr!=NULL)
    {
        if(strcmp(ptr->name,"\0")!=0)
        printf("%s->",ptr->name);
        ptr=ptr->next;
    }
    printf("\n");
}
void delete()
{
    int pos;
    struct Goods*p,*q;
    p=first;
    printf("Enter position : ");
    scanf("%d",&pos);
    if(pos==1)
    {
        first=first->next;
        printf("Deleted Goods : %s\n",p->name);
        free(p);
    }
    else
    {
         for(int i=0;i<pos-1;i++)
    {
        q=p;
        p=p->next;
    }
    q->next=p->next;
    printf("Deleted Goods : %s\n",p->name);
    free(p);
    }
}
```



```
2.Insert new Goods
3.Display Storage list
4.Delete Goods
5.Exit
Enter choice : 2
Enter Goods name : g2
1.Create Storage list
2.Insert new Goods
3.Display Storage list
4.Delete Goods
5.Exit
Enter choice : 2
Enter Goods name : g3
1.Create Storage list
2.Insert new Goods
3.Display Storage list
4.Delete Goods
5.Exit
Enter choice : 3
g1->g2->g3->
1.Create Storage list
2.Insert new Goods
3.Display Storage list
4.Delete Goods
5.Exit
Enter choice : 4
Enter position : 3
Deleted Goods : g3
1.Create Storage list
2.Insert new Goods
3.Display Storage list
4.Delete Goods
5.Exit
Enter choice : 3
g1->g2->
1.Create Storage list
2.Insert new Goods
3.Display Storage list
4.Delete Goods
5.Exit
Enter choice :
```

```
/*Use a linked list to track machine parts inventory.
Operations:
Create a parts inventory list.
Insert a new part.
Delete a part that is used up or obsolete.
Display the current parts inventory.*/
#include<stdlib.h>
#include<stdio.h>
#include<string.h>
struct Parts
{
    char name[20];
    struct Parts *next;
};
struct Parts *first=NULL;
void create();
void insert();
void delete();
void display();
void main()
{
    int choice;
    do
    {
     printf("1.Create Parts inventory \n");
        printf("2.Insert new Parts\n");
        printf("3.Display Partsinventory\n");
        printf("4.Delete Parts\n");
        printf("5.Exit\n");
        printf("Enter choice : ");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
            create();
            break;
            case 2:
            insert();
            break;
            case 3:
```

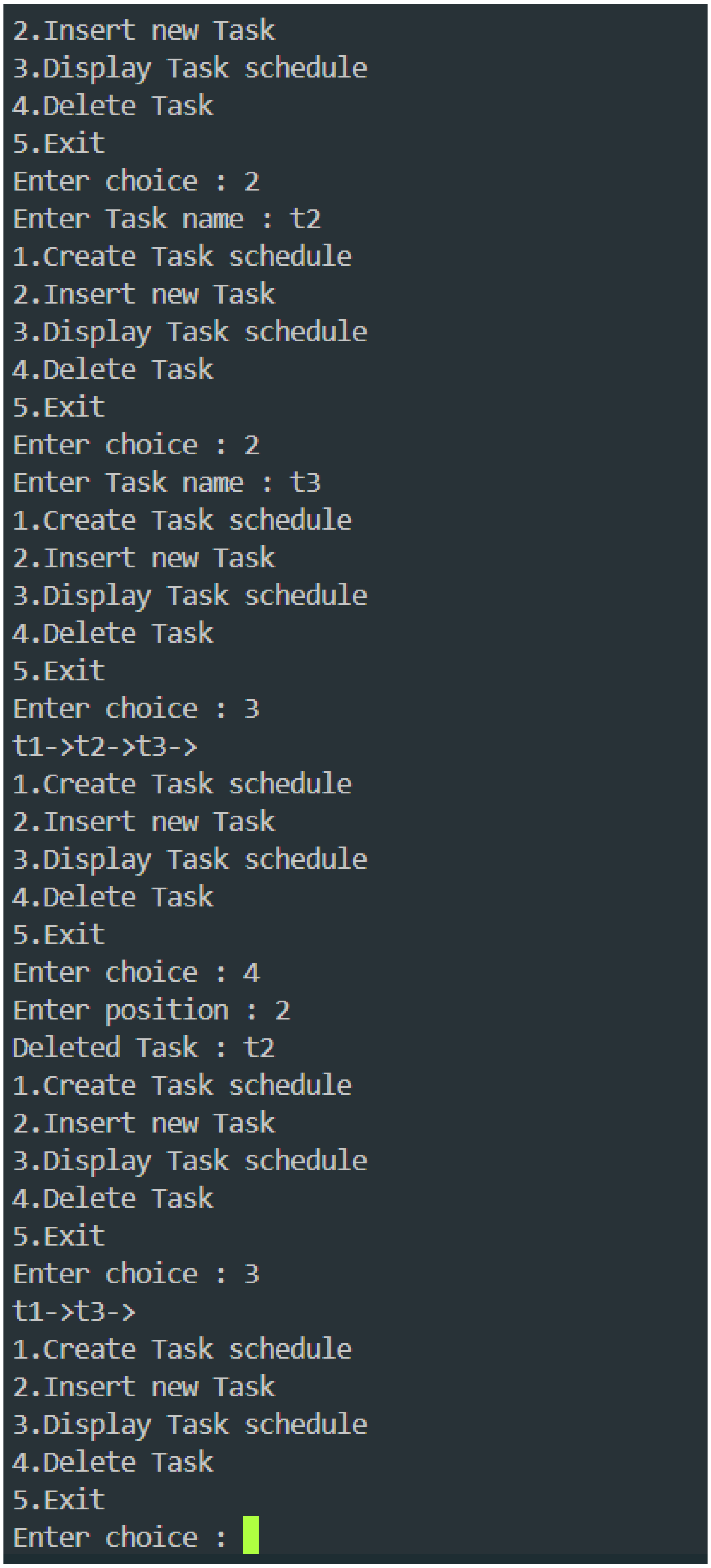
```
            display();
            break;
            case 4:
            delete();
            break;
            case 5:printf("Exiing..\n");
            break;
        }
    } while (choice!=5);

}
void create ()
{
    int n;
    struct Parts *ptr;
    ptr=first;
    printf("Enter size of Inventory: ");
    scanf("%d",&n);
    for(int i=0;i<n;i++)
    {
        struct Parts* newitem=(struct Parts *)malloc(sizeof(struct
Parts));
        newitem->next=NULL;
        strcpy(newitem->name,"\0");
        if(first==NULL)
        {
            first=newitem;
            ptr=first;
        }
        else
        {
            ptr->next=newitem;
            ptr=ptr->next;
        }
    }
}
void insert()
{
    struct Parts *ptr;
    char name[10];
    printf("Enter Parts name : ");
    scanf("%s",name);
    ptr=first;
    while(ptr!=0)
    {
        if(strcmp(ptr->name,"\0")==0)
        {
            strcpy(ptr->name,name);
            return;
        }
        ptr=ptr->next;
    }
}
void display()
{
    struct Parts *ptr;
    ptr=first;
    while(ptr!=NULL)
    {
        if(strcmp(ptr->name,"\0")!=0)
        printf("%s->",ptr->name);
        ptr=ptr->next;
    }
    printf("\n");
}
void delete()
{
    int pos;
    struct Parts*p,*q;
    p=first;
    printf("Enter position : ");
    scanf("%d",&pos);
    if(pos==1)
    {
        first=first->next;
        printf("Deleted Parts : %s\n",p->name);
        free(p);
    }
    else
    {
         for(int i=0;i<pos-1;i++)
    {
        q=p;
        p=p->next;
    }
    q->next=p->next;
    printf("Deleted Parts : %s\n",p->name);
    free(p);
    }
}
```



```
2.Insert new Parts
3.Display Partsinventory
4.Delete Parts
5.Exit
Enter choice : 2
Enter Parts name : p2
1.Create Parts inventory
2.Insert new Parts
3.Display Partsinventory
4.Delete Parts
5.Exit
Enter choice : 2
Enter Parts name : p3
1.Create Parts inventory
2.Insert new Parts
3.Display Partsinventory
4.Delete Parts
5.Exit
Enter choice : 3
p1->p2->p3->
1.Create Parts inventory
2.Insert new Parts
3.Display Partsinventory
4.Delete Parts
5.Exit
Enter choice : 4
Enter position : 1
Deleted Parts : p1
1.Create Parts inventory
2.Insert new Parts
3.Display Partsinventory
4.Delete Parts
5.Exit
Enter choice : 3
p2->p3->
1.Create Parts inventory
2.Insert new Parts
3.Display Partsinventory
4.Delete Parts
5.Exit
Enter choice :
```

```c
/*Manage the schedule of packaging tasks using a linked list.
Operations:
Create a packaging task schedule.
Insert a new packaging task.
Delete a completed packaging task.
Display the current packaging schedule.*/
#include<stdlib.h>
#include<stdio.h>
#include<string.h>
struct Task
{
    char name[20];
    struct Task *next;
};
struct Task *first=NULL;
void create();
void insert();
void delete();
void display();
void main()
{
    int choice;
    do
    {
     printf("1.Create Task schedule \n");
        printf("2.Insert new Task\n");
        printf("3.Display Task schedule\n");
        printf("4.Delete Task\n");
        printf("5.Exit\n");
        printf("Enter choice : ");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
            create();
            break;
            case 2:
            insert();
            break;
            case 3:
```

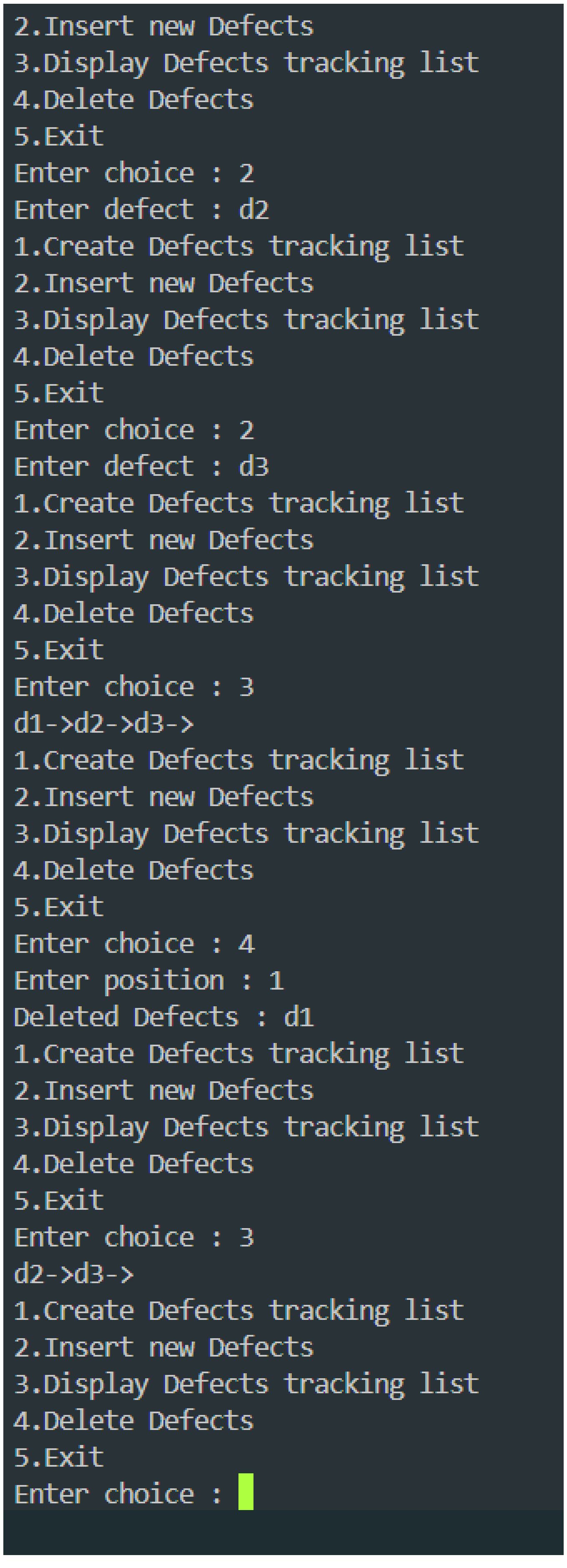
```c
            display();
            break;
            case 4:
            delete();
            break;
            case 5:printf("Exiing..\n");
            break;
        }
    } while (choice!=5);

}
void create ()
{
    int n;
    struct Task *ptr;
    ptr=first;
    printf("Enter size of Task schedule: ");
    scanf("%d",&n);
    for(int i=0;i<n;i++)
    {
        struct Task* newitem=(struct Task *)malloc(sizeof(struct Task));
        newitem->next=NULL;
        strcpy(newitem->name,"\0");
        if(first==NULL)
        {
            first=newitem;
            ptr=first;
        }
        else
        {
            ptr->next=newitem;
            ptr=ptr->next;
        }
    }
}
void insert()
{
    struct Task *ptr;
    char name[10];
    printf("Enter Task name : ");
    scanf("%s",name);
    ptr=first;
    while(ptr!=0)
    {
        if(strcmp(ptr->name,"\0")==0)
        {
            strcpy(ptr->name,name);
            return;
        }
        ptr=ptr->next;
    }
}
void display()
{
    struct Task *ptr;
    ptr=first;
    while(ptr!=NULL)
    {
        if(strcmp(ptr->name,"\0")!=0)
        printf("%s->",ptr->name);
        ptr=ptr->next;
    }
    printf("\n");
}
void delete()
{
    int pos;
    struct Task*p,*q;
    p=first;
    printf("Enter position : ");
    scanf("%d",&pos);
    if(pos==1)
    {
        first=first->next;
        printf("Deleted Task : %s\n",p->name);
        free(p);
    }
    else
    {
         for(int i=0;i<pos-1;i++)
    {
        q=p;
        p=p->next;
    }
    q->next=p->next;
    printf("Deleted Task : %s\n",p->name);
    free(p);
    }
}
```



```
2.Insert new Task
3.Display Task schedule
4.Delete Task
5.Exit
Enter choice : 2
Enter Task name : t2
1.Create Task schedule
2.Insert new Task
3.Display Task schedule
4.Delete Task
5.Exit
Enter choice : 2
Enter Task name : t3
1.Create Task schedule
2.Insert new Task
3.Display Task schedule
4.Delete Task
5.Exit
Enter choice : 3
t1->t2->t3->
1.Create Task schedule
2.Insert new Task
3.Display Task schedule
4.Delete Task
5.Exit
Enter choice : 4
Enter position : 2
Deleted Task : t2
1.Create Task schedule
2.Insert new Task
3.Display Task schedule
4.Delete Task
5.Exit
Enter choice : 3
t1->t3->
1.Create Task schedule
2.Insert new Task
3.Display Task schedule
4.Delete Task
5.Exit
Enter choice : 
```

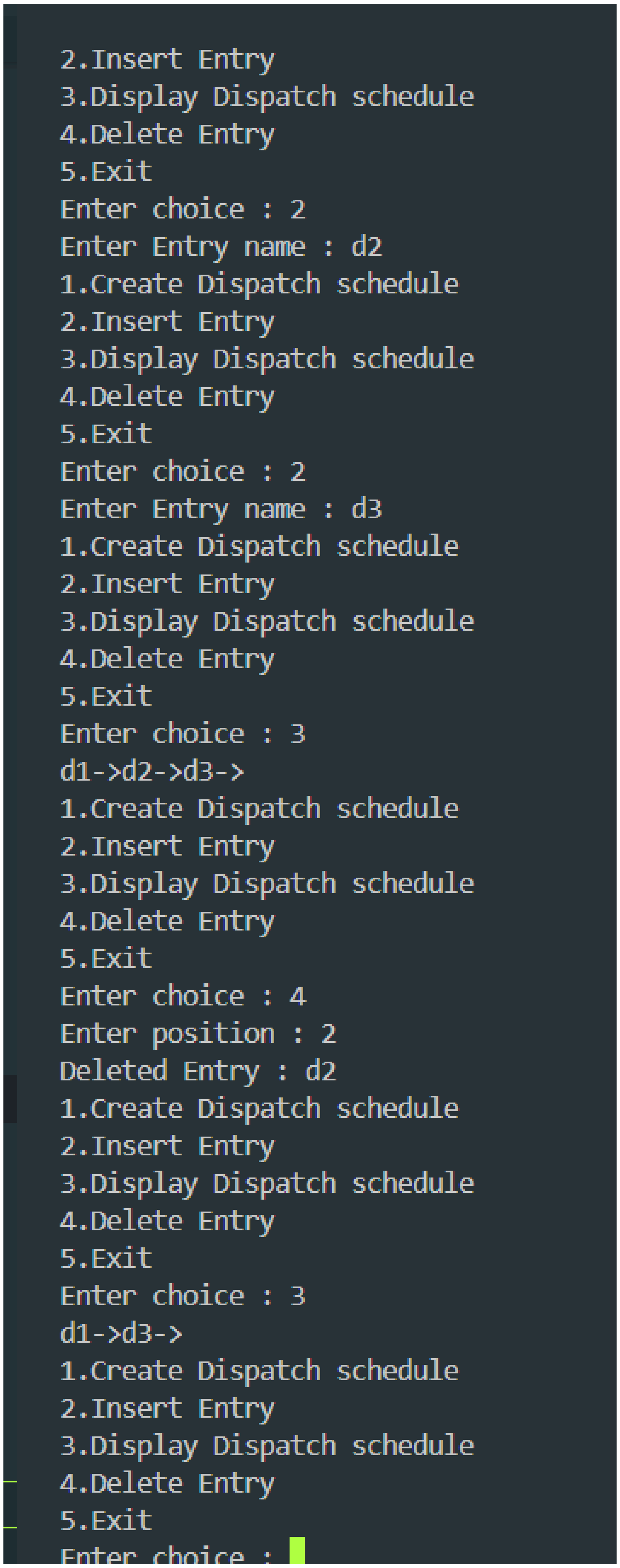
```c
/*Implement a linked list to track defects in the production process.
Operations:
Create a defect tracking list.
Insert a new defect report.
Delete a resolved defect.
Display all current defects.*/
#include<stdlib.h>
#include<stdio.h>
#include<string.h>
struct Defects
{
    char name[20];
    struct Defects *next;
};
struct Defects *first=NULL;
void create();
void insert();
void delete();
void display();
void main()
{
    int choice;
    do
    {
     printf("1.Create Defects tracking list \n");
        printf("2.Insert new Defects\n");
        printf("3.Display Defects tracking list\n");
        printf("4.Delete Defects\n");
        printf("5.Exit\n");
        printf("Enter choice : ");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
            create();
            break;
            case 2:
            insert();
            break;
            case 3:
```

```
            display();
            break;
            case 4:
            delete();
            break;
            case 5:printf("Exiing..\n");
            break;
        }
    } while (choice!=5);

}
void create ()
{
    int n;
    struct Defects *ptr;
    ptr=first;
    printf("Enter size of Defects tracking list : ");
    scanf("%d",&n);
    for(int i=0;i<n;i++)
    {
        struct Defects* newitem=(struct Defects *)malloc(sizeof(struct
Defects));
        newitem->next=NULL;
        strcpy(newitem->name,"\0");
        if(first==NULL)
        {
            first=newitem;
            ptr=first;
        }
        else
        {
            ptr->next=newitem;
            ptr=ptr->next;
        }
    }
}
void insert()
{
    struct Defects *ptr;
    char name[10];
    printf("Enter defect : ");
    scanf("%s",name);
    ptr=first;
    while(ptr!=0)
    {
        if(strcmp(ptr->name,"\0")==0)
        {
            strcpy(ptr->name,name);
            return;
        }
        ptr=ptr->next;
    }
}
void display()
{
    struct Defects *ptr;
    ptr=first;
    while(ptr!=NULL)
    {
        if(strcmp(ptr->name,"\0")!=0)
        printf("%s->",ptr->name);
        ptr=ptr->next;
    }
    printf("\n");
}
void delete()
{
    int pos;
    struct Defects*p,*q;
    p=first;
    printf("Enter position : ");
    scanf("%d",&pos);
    if(pos==1)
    {
        first=first->next;
        printf("Deleted Defects : %s\n",p->name);
        free(p);
    }
    else
    {
          for(int i=0;i<pos-1;i++)
    {
        q=p;
        p=p->next;
    }
    q->next=p->next;
    printf("Deleted Defects : %s\n",p->name);
    free(p);
    }
}
```


```
2.Insert new Defects
3.Display Defects tracking list
4.Delete Defects
5.Exit
Enter choice : 2
Enter defect : d2
1.Create Defects tracking list
2.Insert new Defects
3.Display Defects tracking list
4.Delete Defects
5.Exit
Enter choice : 2
Enter defect : d3
1.Create Defects tracking list
2.Insert new Defects
3.Display Defects tracking list
4.Delete Defects
5.Exit
Enter choice : 3
d1->d2->d3->
1.Create Defects tracking list
2.Insert new Defects
3.Display Defects tracking list
4.Delete Defects
5.Exit
Enter choice : 4
Enter position : 1
Deleted Defects : d1
1.Create Defects tracking list
2.Insert new Defects
3.Display Defects tracking list
4.Delete Defects
5.Exit
Enter choice : 3
d2->d3->
1.Create Defects tracking list
2.Insert new Defects
3.Display Defects tracking list
4.Delete Defects
5.Exit
Enter choice :
```

```
/*Use a linked list to manage the dispatch schedule of finished goods.
Operations:
Create a dispatch schedule.
Insert a new dispatch entry.
Delete a dispatched or canceled entry.
Display the current dispatch schedule.*/
#include<stdlib.h>
#include<stdio.h>
#include<string.h>
struct Entry
{
    char name[20];
    struct Entry *next;
};
struct Entry *first=NULL;
void create();
void insert();
void delete();
void display();
void main()
{
    int choice;
    do
    {
     printf("1.Create Dispatch schedule \n");
        printf("2.Insert Entry\n");
        printf("3.Display Dispatch schedule\n");
        printf("4.Delete Entry\n");
        printf("5.Exit\n");
        printf("Enter choice : ");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
            create();
            break;
            case 2:
            insert();
            break;
            case 3:
```

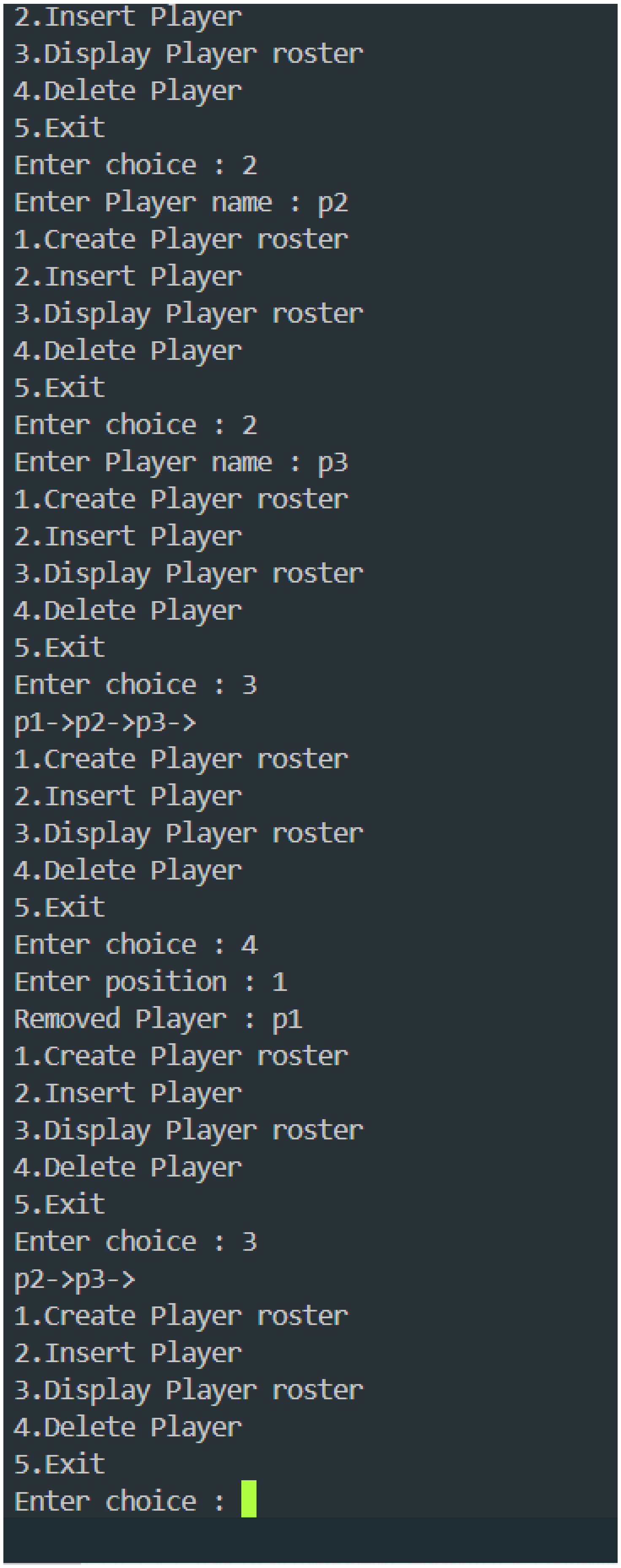
```
            display();
            break;
            case 4:
            delete();
            break;
            case 5:printf("Exiing..\n");
            break;
        }
    } while (choice!=5);

}
void create ()
{
    int n;
    struct Entry *ptr;
    ptr=first;
    printf("Enter size of Dispatch schedule : ");
    scanf("%d",&n);
    for(int i=0;i<n;i++)
    {
        struct Entry* newitem=(struct Entry *)malloc(sizeof(struct
Entry));
        newitem->next=NULL;
        strcpy(newitem->name,"\0");
        if(first==NULL)
        {
            first=newitem;
            ptr=first;
        }
        else
        {
            ptr->next=newitem;
            ptr=ptr->next;
        }
    }
}
void insert()
{
    struct Entry *ptr;
    char name[10];
    printf("Enter Entry name : ");
    scanf("%s",name);
    ptr=first;
    while(ptr!=0)
    {
        if(strcmp(ptr->name,"\0")==0)
        {
            strcpy(ptr->name,name);
            return;
        }
        ptr=ptr->next;
    }
}
void display()
{
    struct Entry *ptr;
    ptr=first;
    while(ptr!=NULL)
    {
        if(strcmp(ptr->name,"\0")!=0)
        printf("%s->",ptr->name);
        ptr=ptr->next;
    }
    printf("\n");
}
void delete()
{
    int pos;
    struct Entry*p,*q;
    p=first;
    printf("Enter position : ");
    scanf("%d",&pos);
    if(pos==1)
    {
        first=first->next;
        printf("Deleted Entry : %s\n",p->name);
        free(p);
    }
    else
    {
         for(int i=0;i<pos-1;i++)
    {
        q=p;
        p=p->next;
    }
    q->next=p->next;
    printf("Deleted Entry : %s\n",p->name);
    free(p);
    }
}
```



```
2.Insert Entry
3.Display Dispatch schedule
4.Delete Entry
5.Exit
Enter choice : 2
Enter Entry name : d2
1.Create Dispatch schedule
2.Insert Entry
3.Display Dispatch schedule
4.Delete Entry
5.Exit
Enter choice : 2
Enter Entry name : d3
1.Create Dispatch schedule
2.Insert Entry
3.Display Dispatch schedule
4.Delete Entry
5.Exit
Enter choice : 3
d1->d2->d3->
1.Create Dispatch schedule
2.Insert Entry
3.Display Dispatch schedule
4.Delete Entry
5.Exit
Enter choice : 4
Enter position : 2
Deleted Entry : d2
1.Create Dispatch schedule
2.Insert Entry
3.Display Dispatch schedule
4.Delete Entry
5.Exit
Enter choice : 3
d1->d3->
1.Create Dispatch schedule
2.Insert Entry
3.Display Dispatch schedule
4.Delete Entry
5.Exit
Enter choice :
```

```c
/*Implement a linked list to manage the roster of players in a sports
team.Operations:
Create a team roster.
Insert a new player.
Delete a player who leaves the team.
Display the current team roster.*/
#include<stdlib.h>
#include<stdio.h>
#include<string.h>
struct Player
{
    char name[20];
    struct Player *next;
};
struct Player *first=NULL;
void create();
void insert();
void delete();
void display();
void main()
{
    int choice;
    do
    {
     printf("1.Create Player roster \n");
        printf("2.Insert Player\n");
        printf("3.Display Player roster\n");
        printf("4.Delete Player\n");
        printf("5.Exit\n");
        printf("Enter choice : ");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
            create();
            break;
            case 2:
            insert();
            break;
            case 3:
```

```
            display();
            break;
            case 4:
            delete();
            break;
            case 5:printf("Exiing..\n");
            break;
        }
    } while (choice!=5);

}
void create ()
{
    int n;
    struct Player *ptr;
    ptr=first;
    printf("Enter size of Player roster : ");
    scanf("%d",&n);
    for(int i=0;i<n;i++)
    {
        struct Player* newitem=(struct Player *)malloc(sizeof(struct
Player));
        newitem->next=NULL;
        strcpy(newitem->name,"\0");
        if(first==NULL)
        {
            first=newitem;
            ptr=first;
        }
        else
        {
            ptr->next=newitem;
            ptr=ptr->next;
        }
    }
}
void insert()
{
    struct Player *ptr;
    char name[10];
    printf("Enter Player name : ");
    scanf("%s",name);
    ptr=first;
    while(ptr!=0)
    {
        if(strcmp(ptr->name,"\0")==0)
        {
            strcpy(ptr->name,name);
            return;
        }
        ptr=ptr->next;
    }
}
void display()
{
    struct Player *ptr;
    ptr=first;
    while(ptr!=NULL)
    {
        if(strcmp(ptr->name,"\0")!=0)
        printf("%s->",ptr->name);
        ptr=ptr->next;
    }
    printf("\n");
}
void delete()
{
    int pos;
    struct Player*p,*q;
    p=first;
    printf("Enter position : ");
    scanf("%d",&pos);
    if(pos==1)
    {
        first=first->next;
        printf("Removed Player : %s\n",p->name);
        free(p);
    }
    else
    {
         for(int i=0;i<pos-1;i++)
    {
        q=p;
        p=p->next;
    }
    q->next=p->next;
    printf("Removed Player : %s\n",p->name);
    free(p);
    }
}
```

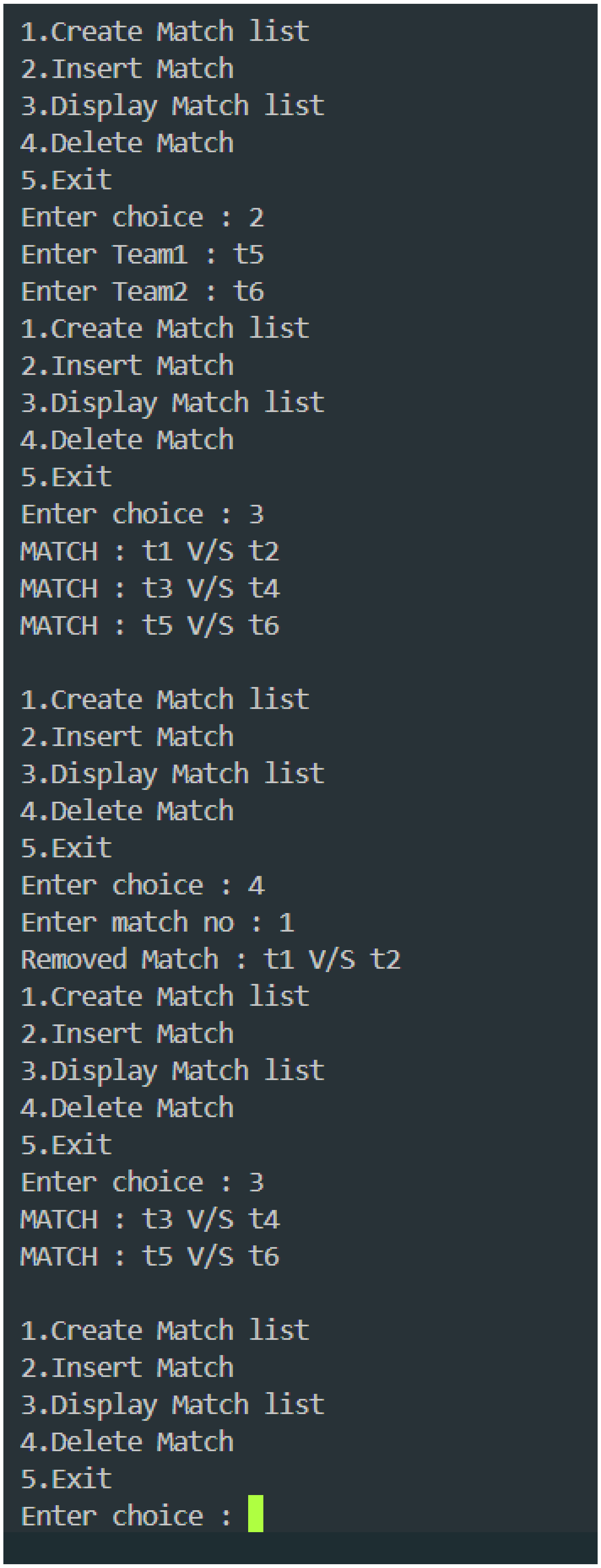


```
2.Insert Player
3.Display Player roster
4.Delete Player
5.Exit
Enter choice : 2
Enter Player name : p2
1.Create Player roster
2.Insert Player
3.Display Player roster
4.Delete Player
5.Exit
Enter choice : 2
Enter Player name : p3
1.Create Player roster
2.Insert Player
3.Display Player roster
4.Delete Player
5.Exit
Enter choice : 3
p1->p2->p3->
1.Create Player roster
2.Insert Player
3.Display Player roster
4.Delete Player
5.Exit
Enter choice : 4
Enter position : 1
Removed Player : p1
1.Create Player roster
2.Insert Player
3.Display Player roster
4.Delete Player
5.Exit
Enter choice : 3
p2->p3->
1.Create Player roster
2.Insert Player
3.Display Player roster
4.Delete Player
5.Exit
Enter choice : 
```

```
/*Use a linked list to schedule matches in a tournament.Operations:
Create a match schedule.
Insert a new match.
Delete a completed or canceled match.
Display the current match schedule.*/
#include<stdlib.h>
#include<stdio.h>
#include<string.h>
struct Match
{
    char Team1[20];
    char Team2[20];
    struct Match *next;
};
struct Match *first=NULL;
void create();
void insert();
void delete();
void display();
void main()
{
    int choice;
    do
    {
     printf("1.Create Match list \n");
        printf("2.Insert Match\n");
        printf("3.Display Match list\n");
        printf("4.Delete Match\n");
        printf("5.Exit\n");
        printf("Enter choice : ");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
            create();
            break;
            case 2:
            insert();
            break;
            case 3:
```

```
            display();
            break;
            case 4:
            delete();
            break;
            case 5:printf("Exiing..\n");
            break;
        }
    } while (choice!=5);

}
void create ()
{
    int n;
    struct Match *ptr;
    ptr=first;
    printf("Enter size of Match list : ");
    scanf("%d",&n);
    for(int i=0;i<n;i++)
    {
        struct Match* newitem=(struct Match *)malloc(sizeof(struct
Match));
        newitem->next=NULL;
        strcpy(newitem->Team1,"\0");
        strcpy(newitem->Team2,"\0");
        if(first==NULL)
        {
            first=newitem;
            ptr=first;
        }
        else
        {
            ptr->next=newitem;
            ptr=ptr->next;
        }
    }
}
void insert()
{
    struct Match *ptr;
```

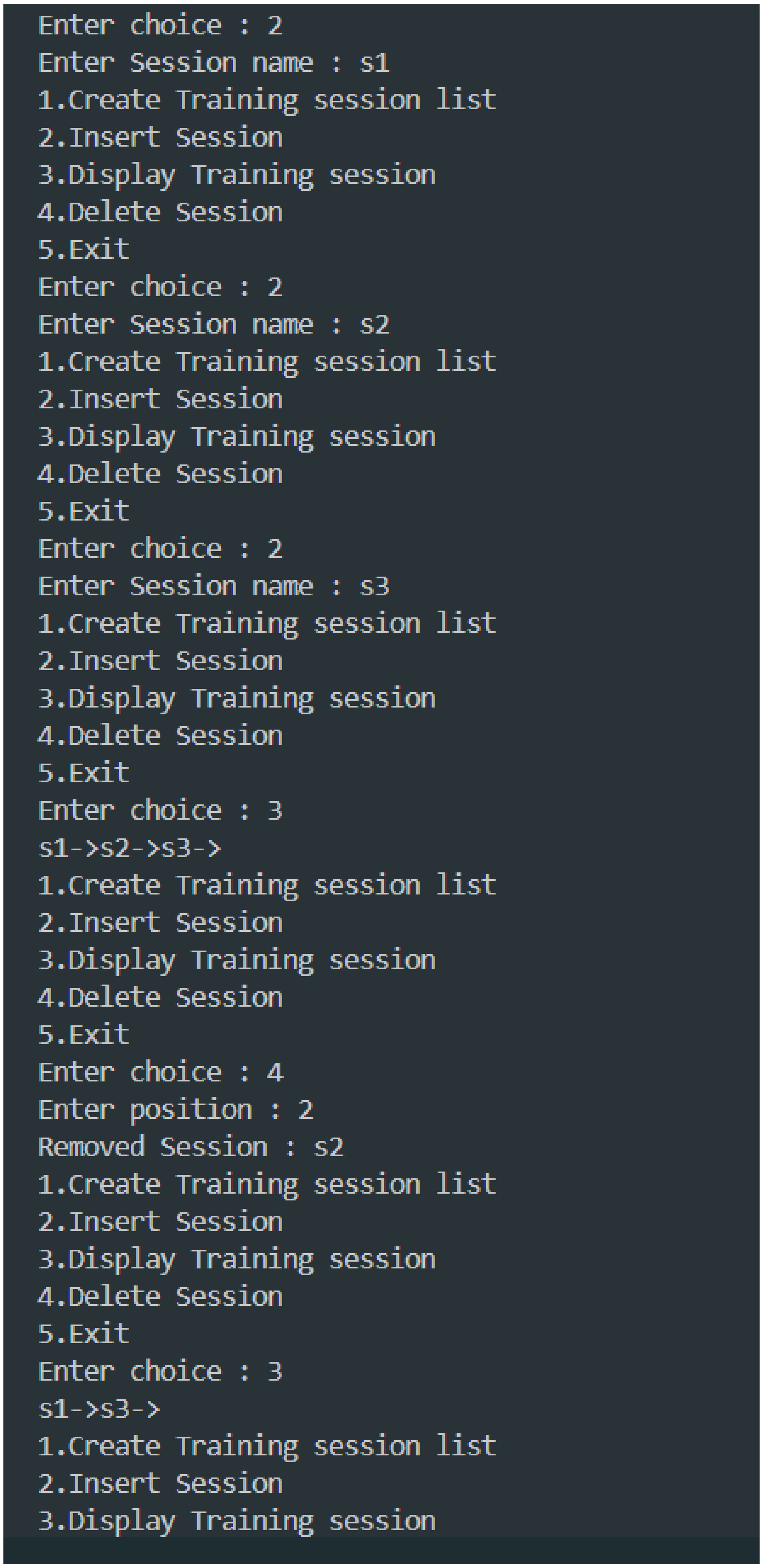
```c
    char name1[20],name2[20];
    printf("Enter Team1 : ");
    scanf("%s",name1);
    printf("Enter Team2 : ");
    scanf("%s",name2);
    ptr=first;
    while(ptr!=0)
    {
        if(strcmp(ptr->Team1,"\0")==0)
        {
            strcpy(ptr->Team1,name1);
            strcpy(ptr->Team2,name2);
            return;
        }
        ptr=ptr->next;
    }
}
void display()
{
    struct Match *ptr;
    ptr=first;
    while(ptr!=NULL)
    {
        if(strcmp(ptr->Team1,"\0")!=0)
        {
            printf("MATCH : %s V/S %s\n",ptr->Team1,ptr->Team2);
        }

        ptr=ptr->next;
    }
    printf("\n");
}
void delete()
{
    int pos;
    struct Match*p,*q;
    p=first;
    printf("Enter match no : ");
    scanf("%d",&pos);
    if(pos==1)
    {
        first=first->next;
        printf("Removed Match : %s V/S %s\n",p->Team1,p->Team2);
        free(p);
    }
    else
    {
         for(int i=0;i<pos-1;i++)
    {
        q=p;
        p=p->next;
    }
    q->next=p->next;
    printf("Removed Match : %s V/S %s\n",p->Team1,p->Team2);
    free(p);
    }
}
```



```
1.Create Match list
2.Insert Match
3.Display Match list
4.Delete Match
5.Exit
Enter choice : 2
Enter Team1 : t5
Enter Team2 : t6
1.Create Match list
2.Insert Match
3.Display Match list
4.Delete Match
5.Exit
Enter choice : 3
MATCH : t1 V/S t2
MATCH : t3 V/S t4
MATCH : t5 V/S t6

1.Create Match list
2.Insert Match
3.Display Match list
4.Delete Match
5.Exit
Enter choice : 4
Enter match no : 1
Removed Match : t1 V/S t2
1.Create Match list
2.Insert Match
3.Display Match list
4.Delete Match
5.Exit
Enter choice : 3
MATCH : t3 V/S t4
MATCH : t5 V/S t6

1.Create Match list
2.Insert Match
3.Display Match list
4.Delete Match
5.Exit
Enter choice :
```

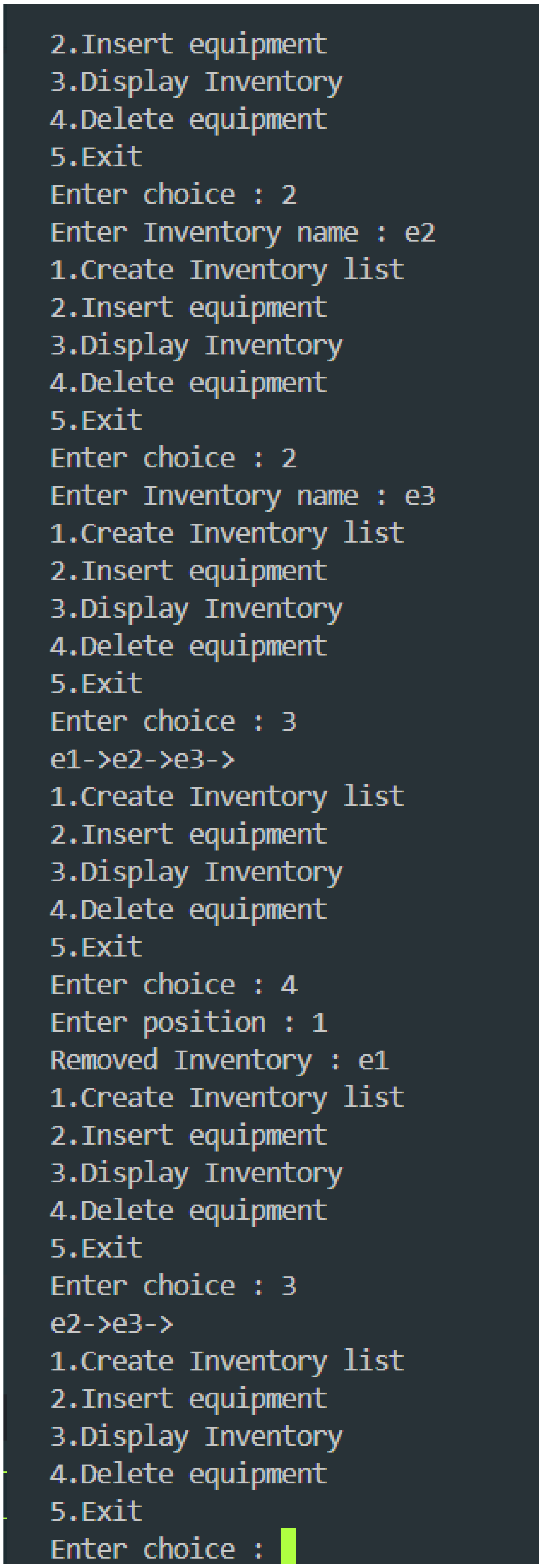
```
/*Develop a linked list to log training sessions for athletes.Operations:
Create a training log.
Insert a new training session.
Delete a completed or canceled session.
Display the training log.*/
#include<stdlib.h>
#include<stdio.h>
#include<string.h>
struct Session
{
    char excercise[20];
    struct Session *next;
};
struct Session *first=NULL;
void create();
void insert();
void delete();
void display();
void main()
{
    int choice;
    do
    {
     printf("1.Create Training session list  \n");
        printf("2.Insert Session\n");
        printf("3.Display Training session\n");
        printf("4.Delete Session\n");
        printf("5.Exit\n");
        printf("Enter choice : ");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
            create();
            break;
            case 2:
            insert();
            break;
            case 3:
            display();
```

```c
            break;
            case 4:
            delete();
            break;
            case 5:printf("Exiing..\n");
            break;
        }
    } while (choice!=5);

}
void create ()
{
    int n;
    struct Session *ptr;
    ptr=first;
    printf("Enter size of Training session list : ");
    scanf("%d",&n);
    for(int i=0;i<n;i++)
    {
        struct Session* newitem=(struct Session *)malloc(sizeof(struct
Session));
        newitem->next=NULL;
        strcpy(newitem->excercise,"\0");
        if(first==NULL)
        {
            first=newitem;
            ptr=first;
        }
        else
        {
            ptr->next=newitem;
            ptr=ptr->next;
        }
    }
}
void insert()
{
    struct Session *ptr;
    char name[10];
    printf("Enter Session name : ");
    scanf("%s",name);
    ptr=first;
    while(ptr!=0)
    {
        if(strcmp(ptr->excercise,"\0")==0)
        {
            strcpy(ptr->excercise,name);
            return;
        }
        ptr=ptr->next;
    }
}
void display()
{
    struct Session *ptr;
    ptr=first;
    while(ptr!=NULL)
    {
        if(strcmp(ptr->excercise,"\0")!=0)
        printf("%s->",ptr->excercise);
        ptr=ptr->next;
    }
    printf("\n");
}
void delete()
{
    int pos;
    struct Session*p,*q;
    p=first;
    printf("Enter position : ");
    scanf("%d",&pos);
    if(pos==1)
    {
        first=first->next;
        printf("Removed Session : %s\n",p->excercise);
        free(p);
    }
    else
    {
         for(int i=0;i<pos-1;i++)
    {
        q=p;
        p=p->next;
    }
    q->next=p->next;
    printf("Removed Session : %s\n",p->excercise);
    free(p);
    }
}
```


```
Enter choice : 2
Enter Session name : s1
1.Create Training session list
2.Insert Session
3.Display Training session
4.Delete Session
5.Exit
Enter choice : 2
Enter Session name : s2
1.Create Training session list
2.Insert Session
3.Display Training session
4.Delete Session
5.Exit
Enter choice : 2
Enter Session name : s3
1.Create Training session list
2.Insert Session
3.Display Training session
4.Delete Session
5.Exit
Enter choice : 3
s1->s2->s3->
1.Create Training session list
2.Insert Session
3.Display Training session
4.Delete Session
5.Exit
Enter choice : 4
Enter position : 2
Removed Session : s2
1.Create Training session list
2.Insert Session
3.Display Training session
4.Delete Session
5.Exit
Enter choice : 3
s1->s3->
1.Create Training session list
2.Insert Session
3.Display Training session
```

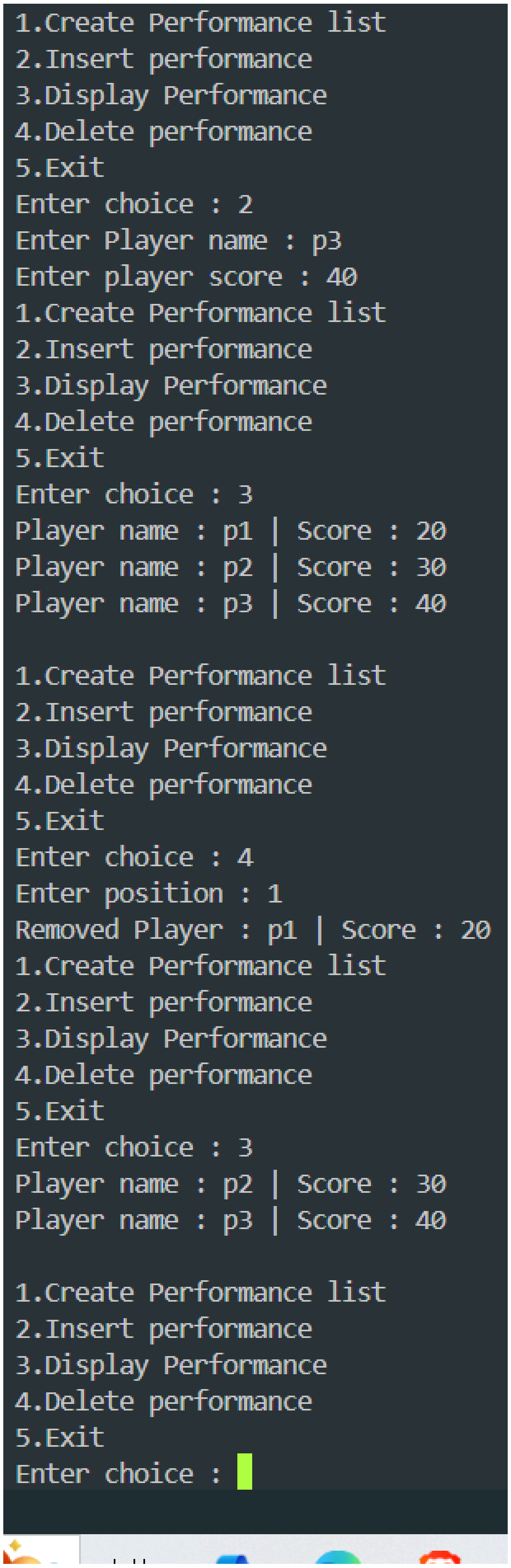
```c
/*Use a linked list to manage the inventory of sports
equipment.Operations:
Create an equipment inventory list.
Insert a new equipment item.
Delete an item that is no longer usable.
Display the current equipment inventory.*/
#include<stdlib.h>
#include<stdio.h>
#include<string.h>
struct Inventory
{
    char equipment[20];
    struct Inventory *next;
};
struct Inventory *first=NULL;
void create();
void insert();
void delete();
void display();
void main()
{
    int choice;
    do
    {
     printf("1.Create Inventory list  \n");
        printf("2.Insert equipment\n");
        printf("3.Display Inventory\n");
        printf("4.Delete equipment\n");
        printf("5.Exit\n");
        printf("Enter choice : ");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
            create();
            break;
            case 2:
            insert();
            break;
            case 3:
```

```
            display();
            break;
            case 4:
            delete();
            break;
            case 5:printf("Exiing..\n");
            break;
        }
    } while (choice!=5);

}
void create ()
{
    int n;
    struct Inventory *ptr;
    ptr=first;
    printf("Enter size of Inventory list : ");
    scanf("%d",&n);
    for(int i=0;i<n;i++)
    {
        struct Inventory* newitem=(struct Inventory *)malloc(sizeof(struct
Inventory));
        newitem->next=NULL;
        strcpy(newitem->equipment,"\0");
        if(first==NULL)
        {
            first=newitem;
            ptr=first;
        }
        else
        {
            ptr->next=newitem;
            ptr=ptr->next;
        }
    }
}
void insert()
{
    struct Inventory *ptr;
    char name[10];
    printf("Enter Inventory name : ");
    scanf("%s",name);
    ptr=first;
    while(ptr!=0)
    {
        if(strcmp(ptr->equipment,"\0")==0)
        {
            strcpy(ptr->equipment,name);
            return;
        }
        ptr=ptr->next;
    }
}
void display()
{
    struct Inventory *ptr;
    ptr=first;
    while(ptr!=NULL)
    {
        if(strcmp(ptr->equipment,"\0")!=0)
        printf("%s->",ptr->equipment);
        ptr=ptr->next;
    }
    printf("\n");
}
void delete()
{
    int pos;
    struct Inventory*p,*q;
    p=first;
    printf("Enter position : ");
    scanf("%d",&pos);
    if(pos==1)
    {
        first=first->next;
        printf("Removed Inventory : %s\n",p->equipment);
        free(p);
    }
    else
    {
         for(int i=0;i<pos-1;i++)
    {
        q=p;
        p=p->next;
    }
    q->next=p->next;
    printf("Removed Inventory : %s\n",p->equipment);
    free(p);
    }
}
```



```
2.Insert equipment
3.Display Inventory
4.Delete equipment
5.Exit
Enter choice : 2
Enter Inventory name : e2
1.Create Inventory list
2.Insert equipment
3.Display Inventory
4.Delete equipment
5.Exit
Enter choice : 2
Enter Inventory name : e3
1.Create Inventory list
2.Insert equipment
3.Display Inventory
4.Delete equipment
5.Exit
Enter choice : 3
e1->e2->e3->
1.Create Inventory list
2.Insert equipment
3.Display Inventory
4.Delete equipment
5.Exit
Enter choice : 4
Enter position : 1
Removed Inventory : e1
1.Create Inventory list
2.Insert equipment
3.Display Inventory
4.Delete equipment
5.Exit
Enter choice : 3
e2->e3->
1.Create Inventory list
2.Insert equipment
3.Display Inventory
4.Delete equipment
5.Exit
Enter choice :
```

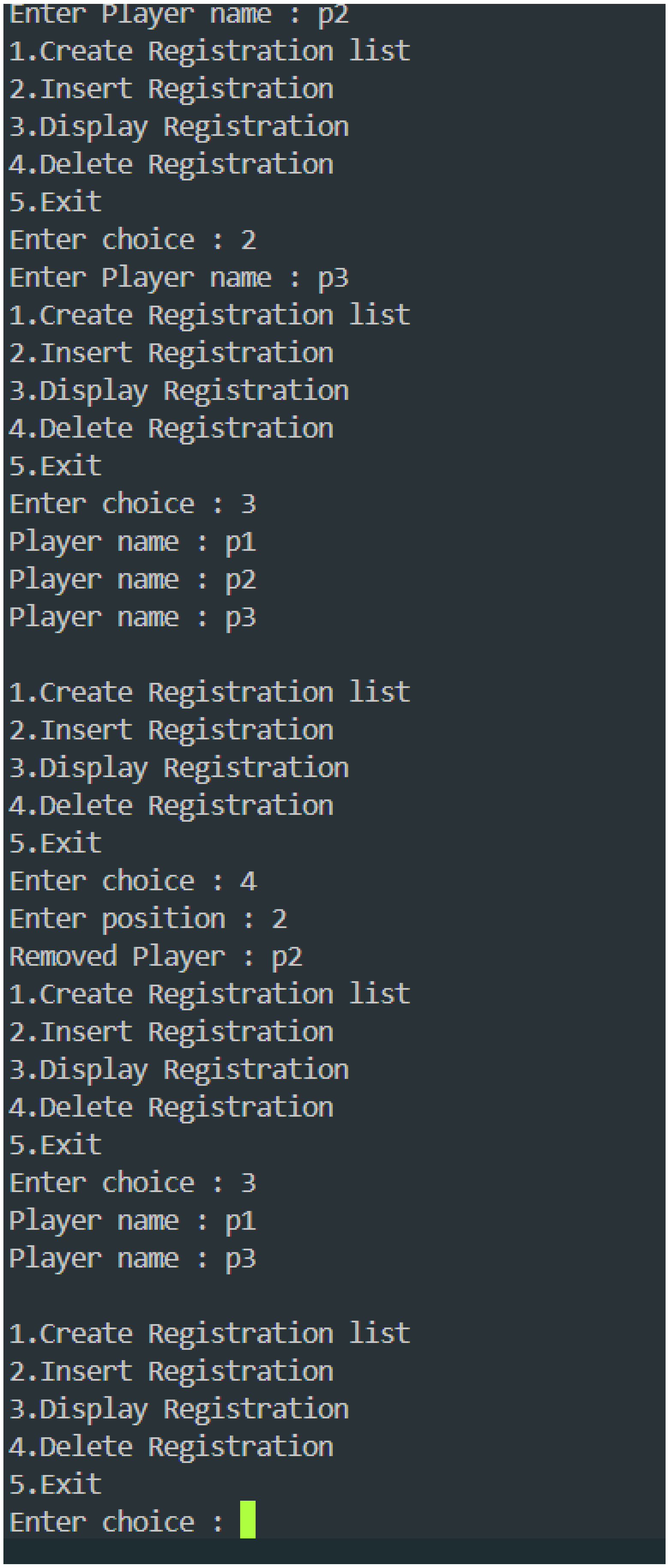
```c
/*Implement a linked list to track player performance over the
season.Operations:
Create a performance record list.
Insert a new performance entry.
Delete an outdated or erroneous entry.
Display all performance records.*/
#include<stdlib.h>
#include<stdio.h>
#include<string.h>
struct Performance
{
    char name[20];
    int score;
    struct Performance *next;
};
struct Performance *first=NULL;
void create();
void insert();
void delete();
void display();
void main()
{
    int choice;
    do
    {
     printf("1.Create Performance list  \n");
        printf("2.Insert performance\n");
        printf("3.Display Performance\n");
        printf("4.Delete performance\n");
        printf("5.Exit\n");
        printf("Enter choice : ");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
            create();
            break;
            case 2:
            insert();
            break;
            case 3:
            display();
            break;
            case 4:
            delete();
            break;
            case 5:printf("Exiing..\n");
            break;
        }
    } while (choice!=5);

}
void create ()
{
    int n;
    struct Performance *ptr;
    ptr=first;
    printf("Enter size of Performance list : ");
    scanf("%d",&n);
    for(int i=0;i<n;i++)
    {
        struct Performance* newitem=(struct Performance
*)malloc(sizeof(struct Performance));
        newitem->next=NULL;
        strcpy(newitem->name,"\0");
        newitem->score=-1;
        if(first==NULL)
        {
            first=newitem;
            ptr=first;
        }
        else
        {
            ptr->next=newitem;
            ptr=ptr->next;
        }
    }
}
void insert()
{
    struct Performance *ptr;
    char name[10];
    int s;
    printf("Enter Player name : ");
    scanf("%s",name);
    printf("Enter player score : ");
    scanf("%d",&s);
    ptr=first;
    while(ptr!=0)
    {
        if(strcmp(ptr->name,"\0")==0)
        {
            strcpy(ptr->name,name);
            ptr->score=s;
            return;
        }
        ptr=ptr->next;
    }
}
void display()
{
    struct Performance *ptr;
    ptr=first;
    while(ptr!=NULL)
    {

        if(strcmp(ptr->name,"\0")!=0)
        printf("Player name : %s | Score : %d\n",ptr->name,ptr->score);
        ptr=ptr->next;
    }
    printf("\n");
}
void delete()
{
    int pos;
    struct Performance*p,*q;
    p=first;
    printf("Enter position : ");
    scanf("%d",&pos);
    if(pos==1)
    {
        first=first->next;
        printf("Removed Player : %s | Score : %d\n",p->name,p->score);
        free(p);
    }
    else
    {
         for(int i=0;i<pos-1;i++)
    {
        q=p;
        p=p->next;
    }
    q->next=p->next;
     printf("Removed Player : %s | Score : %d\n",p->name,p->score);
    free(p);
    }
}
```



```
1.Create Performance list
2.Insert performance
3.Display Performance
4.Delete performance
5.Exit
Enter choice : 2
Enter Player name : p3
Enter player score : 40
1.Create Performance list
2.Insert performance
3.Display Performance
4.Delete performance
5.Exit
Enter choice : 3
Player name : p1 | Score : 20
Player name : p2 | Score : 30
Player name : p3 | Score : 40

1.Create Performance list
2.Insert performance
3.Display Performance
4.Delete performance
5.Exit
Enter choice : 4
Enter position : 1
Removed Player : p1 | Score : 20
1.Create Performance list
2.Insert performance
3.Display Performance
4.Delete performance
5.Exit
Enter choice : 3
Player name : p2 | Score : 30
Player name : p3 | Score : 40

1.Create Performance list
2.Insert performance
3.Display Performance
4.Delete performance
5.Exit
Enter choice :
```

```c
/*Use a linked list to manage athlete registrations for sports
events.Operations:
Create a registration list.
Insert a new registration.
Delete a canceled registration.
Display all current registrations*/
#include<stdlib.h>
#include<stdio.h>
#include<string.h>
struct Registration
{
    char name[20];
    struct Registration *next;
};
struct Registration *first=NULL;
void create();
void insert();
void delete();
void display();
void main()
{
    int choice;
    do
    {
     printf("1.Create Registration list  \n");
        printf("2.Insert Registration\n");
        printf("3.Display Registration\n");
        printf("4.Delete Registration\n");
        printf("5.Exit\n");
        printf("Enter choice : ");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
            create();
            break;
            case 2:
            insert();
            break;
            case 3:
```

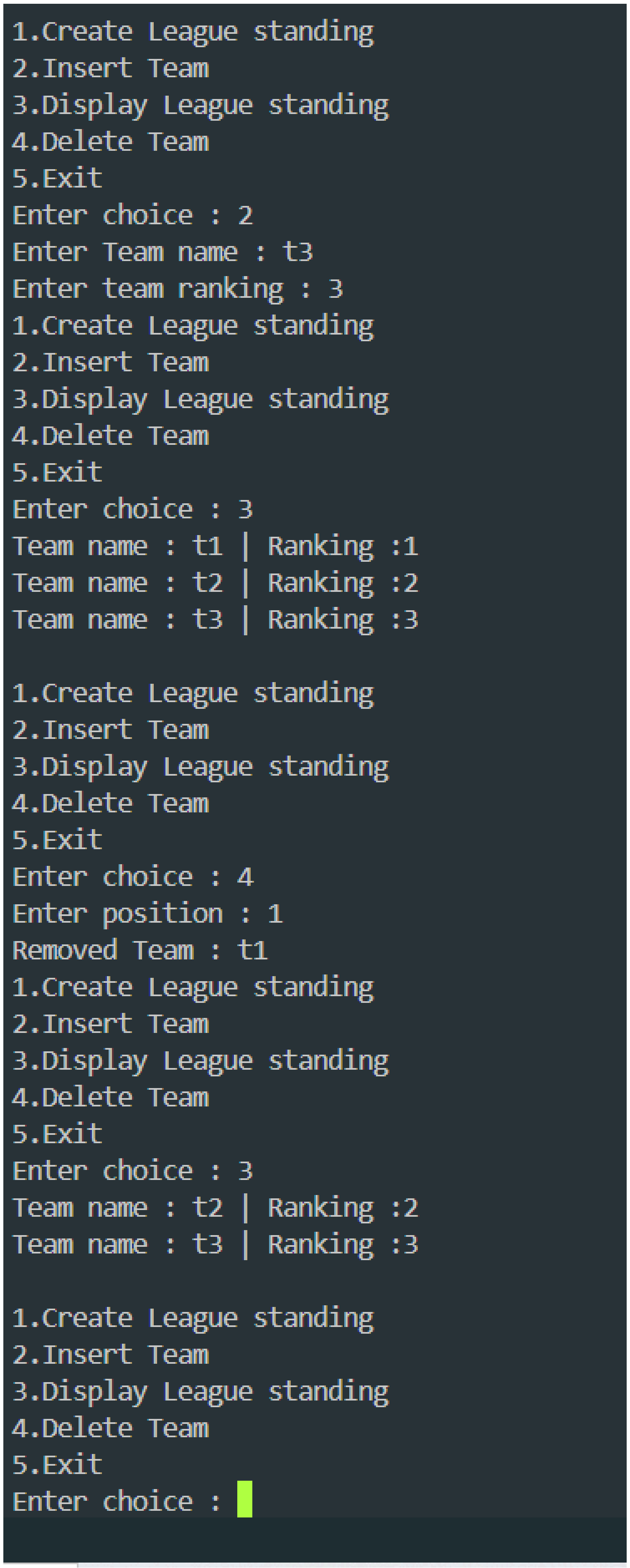
```
            display();
            break;
            case 4:
            delete();
            break;
            case 5:printf("Exiing..\n");
            break;
        }
    } while (choice!=5);

}
void create ()
{
    int n;
    struct Registration *ptr;
    ptr=first;
    printf("Enter size of Registration list : ");
    scanf("%d",&n);
    for(int i=0;i<n;i++)
    {
        struct Registration* newitem=(struct Registration
*)malloc(sizeof(struct Registration));
        newitem->next=NULL;
        strcpy(newitem->name,"\0");
        if(first==NULL)
        {
            first=newitem;
            ptr=first;
        }
        else
        {
            ptr->next=newitem;
            ptr=ptr->next;
        }
    }
}
void insert()
{
    struct Registration *ptr;
    char name[10];
    printf("Enter Player name : ");
    scanf("%s",name);
    ptr=first;
    while(ptr!=0)
    {
        if(strcmp(ptr->name,"\0")==0)
        {
            strcpy(ptr->name,name);
            return;
        }
        ptr=ptr->next;
    }
}
void display()
{
    struct Registration *ptr;
    ptr=first;
    while(ptr!=NULL)
    {

        if(strcmp(ptr->name,"\0")!=0)
        printf("Player name : %s\n",ptr->name);
        ptr=ptr->next;
    }
    printf("\n");
}
void delete()
{
    int pos;
    struct Registration*p,*q;
    p=first;
    printf("Enter position : ");
    scanf("%d",&pos);
    if(pos==1)
    {
        first=first->next;
        printf("Removed Player : %s\n",p->name);
        free(p);
    }
    else
    {
         for(int i=0;i<pos-1;i++)
    {
        q=p;
        p=p->next;
    }
    q->next=p->next;
    printf("Removed Player : %s\n",p->name);
    free(p);
    }
}
```



```
Enter Player name : p2
1.Create Registration list
2.Insert Registration
3.Display Registration
4.Delete Registration
5.Exit
Enter choice : 2
Enter Player name : p3
1.Create Registration list
2.Insert Registration
3.Display Registration
4.Delete Registration
5.Exit
Enter choice : 3
Player name : p1
Player name : p2
Player name : p3

1.Create Registration list
2.Insert Registration
3.Display Registration
4.Delete Registration
5.Exit
Enter choice : 4
Enter position : 2
Removed Player : p2
1.Create Registration list
2.Insert Registration
3.Display Registration
4.Delete Registration
5.Exit
Enter choice : 3
Player name : p1
Player name : p3

1.Create Registration list
2.Insert Registration
3.Display Registration
4.Delete Registration
5.Exit
Enter choice :
```

```
/*Develop a linked list to manage the standings of teams in a sports
league.Operations:
Create a league standings list.
Insert a new team.
Delete a team that withdraws.
Display the current league standings.*/
#include<stdlib.h>
#include<stdio.h>
#include<string.h>
struct League
{
    char name[20];
    int rank;
    struct League *next;
};
struct League *first=NULL;
void create();
void insert();
void delete();
void display();
void main()
{
    int choice;
    do
    {
     printf("1.Create League standing  \n");
        printf("2.Insert Team\n");
        printf("3.Display League standing\n");
        printf("4.Delete Team\n");
        printf("5.Exit\n");
        printf("Enter choice : ");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
            create();
            break;
            case 2:
            insert();
            break;
            case 3:
            display();
            break;
            case 4:
            delete();
            break;
            case 5:printf("Exiing..\n");
            break;
        }
    } while (choice!=5);

}
void create ()
{
    int n;
    struct League *ptr;
    ptr=first;
    printf("Enter size of League standing : ");
    scanf("%d",&n);
    for(int i=0;i<n;i++)
    {
        struct League* newitem=(struct League *)malloc(sizeof(struct
League));
        newitem->next=NULL;
        strcpy(newitem->name,"\0");
        newitem->rank=0;
        if(first==NULL)
        {
            first=newitem;
            ptr=first;
        }
        else
        {
            ptr->next=newitem;
            ptr=ptr->next;
        }
    }
}
void insert()
{
```

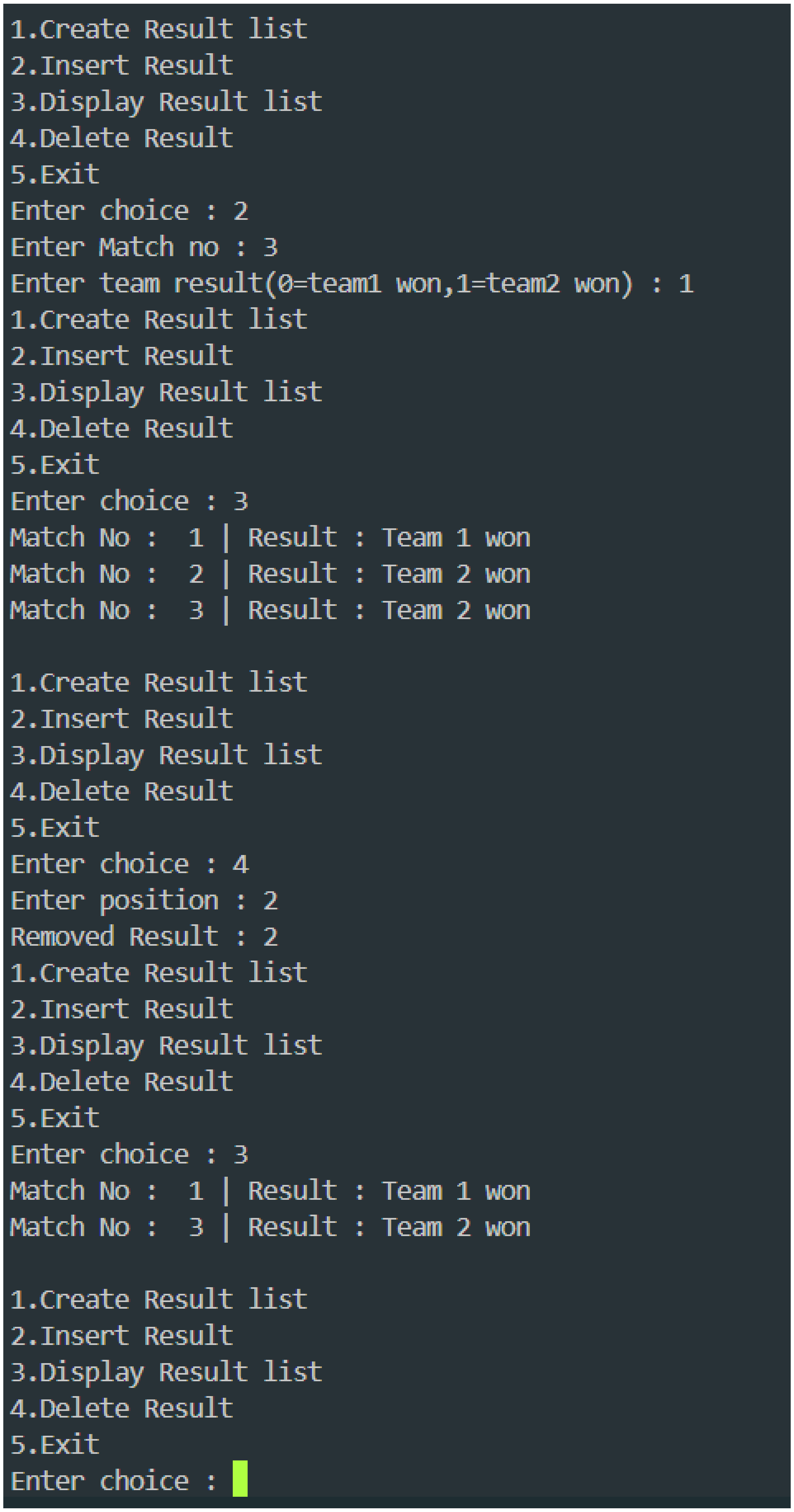
```c
    struct League *ptr;
    char name[10];
    int r;
    printf("Enter Team name : ");
    scanf("%s",name);
    printf("Enter team ranking : ");
    scanf("%d",&r);
    ptr=first;
    while(ptr!=0)
    {
        if(strcmp(ptr->name,"\0")==0)
        {
            strcpy(ptr->name,name);
            ptr->rank=r;
            return;
        }
        ptr=ptr->next;
    }
}
void display()
{
    struct League *ptr;
    ptr=first;
    while(ptr!=NULL)
    {

        if(strcmp(ptr->name,"\0")!=0)
        printf("Team name : %s | Ranking :%d \n",ptr->name,ptr->rank);
        ptr=ptr->next;
    }
    printf("\n");
}
void delete()
{
    int pos;
    struct League*p,*q;
    p=first;
    printf("Enter position : ");
    scanf("%d",&pos);
    if(pos==1)
    {
        first=first->next;
        printf("Removed Team : %s\n",p->name);
        free(p);
    }
    else
    {
         for(int i=0;i<pos-1;i++)
    {
        q=p;
        p=p->next;
    }
    q->next=p->next;
    printf("Removed Team : %s\n",p->name);
    free(p);
    }
}
```



```
1.Create League standing
2.Insert Team
3.Display League standing
4.Delete Team
5.Exit
Enter choice : 2
Enter Team name : t3
Enter team ranking : 3
1.Create League standing
2.Insert Team
3.Display League standing
4.Delete Team
5.Exit
Enter choice : 3
Team name : t1 | Ranking :1
Team name : t2 | Ranking :2
Team name : t3 | Ranking :3

1.Create League standing
2.Insert Team
3.Display League standing
4.Delete Team
5.Exit
Enter choice : 4
Enter position : 1
Removed Team : t1
1.Create League standing
2.Insert Team
3.Display League standing
4.Delete Team
5.Exit
Enter choice : 3
Team name : t2 | Ranking :2
Team name : t3 | Ranking :3

1.Create League standing
2.Insert Team
3.Display League standing
4.Delete Team
5.Exit
Enter choice : 
```

```
/*Implement a linked list to record results of matches.Operations:
Create a match result list.
Insert a new match result.
Delete an incorrect or outdated result.
Display all recorded match results.*/
#include<stdlib.h>
#include<stdio.h>
#include<string.h>
struct Match
{
    int match;
    int result;
    struct Match *next;
};
struct Match *first=NULL;
void create();
void insert();
void delete();
void display();
void main()
{
    int choice;
    do
    {
     printf("1.Create Result list  \n");
        printf("2.Insert Result\n");
        printf("3.Display Result list\n");
        printf("4.Delete Result\n");
        printf("5.Exit\n");
        printf("Enter choice : ");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
            create();
            break;
            case 2:
            insert();
            break;
            case 3:
```

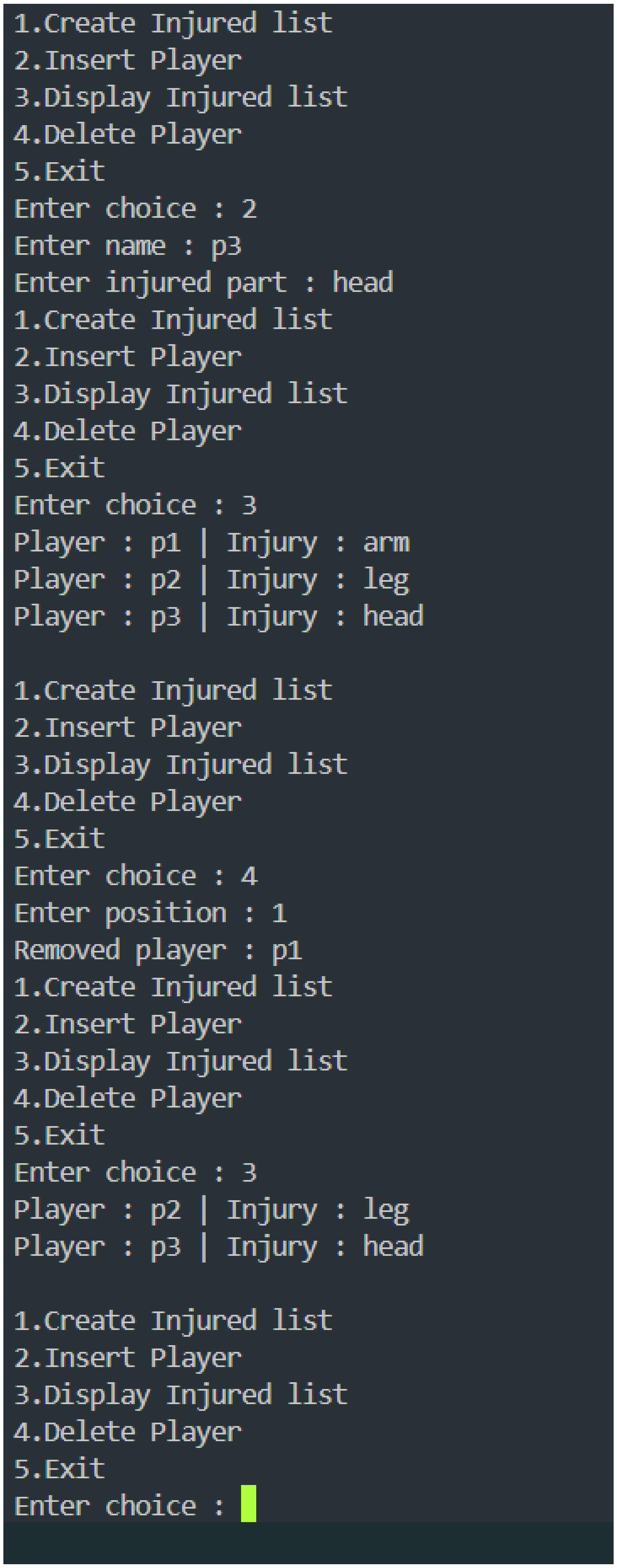
```
            display();
            break;
            case 4:
            delete();
            break;
            case 5:printf("Exiing..\n");
            break;
        }
    } while (choice!=5);

}
void create ()
{
    int n;
    struct Match *ptr;
    ptr=first;
    printf("Enter size of Match list: ");
    scanf("%d",&n);
    for(int i=0;i<n;i++)
    {
        struct Match* newitem=(struct Match *)malloc(sizeof(struct
Match));
        newitem->next=NULL;
        newitem->match=-1;
        newitem->result=-1;
        if(first==NULL)
        {
            first=newitem;
            ptr=first;
        }
        else
        {
            ptr->next=newitem;
            ptr=ptr->next;
        }
    }
}
void insert()
{
    struct Match *ptr;
    int r,m;
    printf("Enter Match no : ");
    scanf("%d",&m);
    printf("Enter team result(0=team1 won,1=team2 won) : ");
    scanf("%d",&r);
    ptr=first;
    while(ptr!=0)
    {
        if(ptr->match==-1)
        {
            ptr->match=m;
            ptr->result=r;
            return;
        }
        ptr=ptr->next;
    }
}
void display()
{
    struct Match *ptr;
    ptr=first;
    while(ptr!=NULL)
    {

        if(ptr->match!=(-1))
        {
            if(ptr->result==0)
            printf("Match No :  %d | Result : Team 1 won \n",ptr->match);
            else if(ptr->result==1)
            printf("Match No :  %d | Result : Team 2 won \n",ptr->match);
        }

        ptr=ptr->next;
    }
    printf("\n");
}
void delete()
{
    int pos;
    struct Match*p,*q;
    p=first;
    printf("Enter position : ");
    scanf("%d",&pos);
    if(pos==1)
    {
        first=first->next;
        printf("Removed Result : %d\n",p->match);
        free(p);
    }
    else
    {
         for(int i=0;i<pos-1;i++)
    {
        q=p;
        p=p->next;
    }
    q->next=p->next;
    printf("Removed Result : %d\n",p->match);
    free(p);
    }
}
```



```
1.Create Result list
2.Insert Result
3.Display Result list
4.Delete Result
5.Exit
Enter choice : 2
Enter Match no : 3
Enter team result(0=team1 won,1=team2 won) : 1
1.Create Result list
2.Insert Result
3.Display Result list
4.Delete Result
5.Exit
Enter choice : 3
Match No :  1 | Result : Team 1 won
Match No :  2 | Result : Team 2 won
Match No :  3 | Result : Team 2 won

1.Create Result list
2.Insert Result
3.Display Result list
4.Delete Result
5.Exit
Enter choice : 4
Enter position : 2
Removed Result : 2
1.Create Result list
2.Insert Result
3.Display Result list
4.Delete Result
5.Exit
Enter choice : 3
Match No :  1 | Result : Team 1 won
Match No :  3 | Result : Team 2 won

1.Create Result list
2.Insert Result
3.Display Result list
4.Delete Result
5.Exit
Enter choice : 
```

```c
/*Use a linked list to track injuries of players.Operations:
Create an injury tracker list.
Insert a new injury report.
Delete a resolved or erroneous injury report.
Display all current injury reports.*/
#include<stdlib.h>
#include<stdio.h>
#include<string.h>
struct Injury
{
    char name[20];
    char part[20];
    struct Injury *next;
};
struct Injury *first=NULL;
void create();
void insert();
void delete();
void display();
void main()
{
    int choice;
    do
    {
     printf("1.Create Injured list  \n");
        printf("2.Insert Player\n");
        printf("3.Display Injured list\n");
        printf("4.Delete Player\n");
        printf("5.Exit\n");
        printf("Enter choice : ");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
            create();
            break;
            case 2:
            insert();
            break;
            case 3:
```

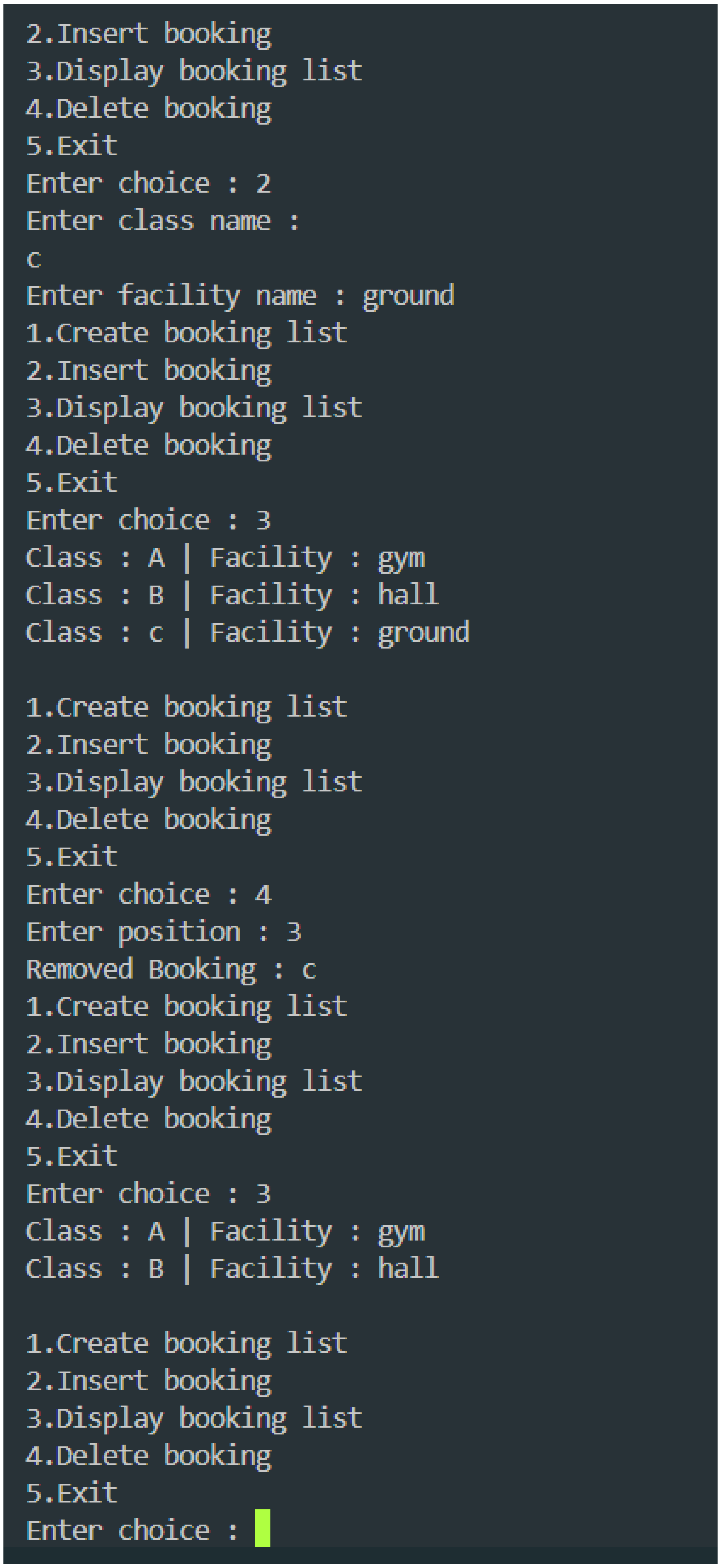
```
            display();
            break;
            case 4:
            delete();
            break;
            case 5:printf("Exiing..\n");
            break;
        }
    } while (choice!=5);

}
void create ()
{
    int n;
    struct Injury *ptr;
    ptr=first;
    printf("Enter size of Injury list: ");
    scanf("%d",&n);
    for(int i=0;i<n;i++)
    {
        struct Injury* newitem=(struct Injury *)malloc(sizeof(struct
Injury));
        newitem->next=NULL;
        strcpy(newitem->name,"\0");
        strcpy(newitem->part,"\0");
        if(first==NULL)
        {
            first=newitem;
            ptr=first;
        }
        else
        {
            ptr->next=newitem;
            ptr=ptr->next;
        }
    }
}
void insert()
{
    struct Injury *ptr;
    char name[20],part[20];
    printf("Enter name : ");
    scanf("%s",name);
   printf("Enter injured part : ");
    scanf("%s",part);
    ptr=first;
    while(ptr!=0)
    {
        if(strcmp(ptr->name,"\0")==0)
        {
            strcpy(ptr->name,name);
            strcpy(ptr->part,part);
            return;
        }
        ptr=ptr->next;
    }
}
void display()
{
    struct Injury *ptr;
    ptr=first;
    while(ptr!=NULL)
    {

        if(strcmp(ptr->name,"\0")!=0)
        {
            printf("Player : %s | Injury : %s\n",ptr->name,ptr->part);
        }

        ptr=ptr->next;
    }
    printf("\n");
}
void delete()
{
    int pos;
    struct Injury*p,*q;
    p=first;
    printf("Enter position : ");
    scanf("%d",&pos);
    if(pos==1)
    {
        first=first->next;
        printf("Removed player : %s\n",p->name);
        free(p);
    }
    else
    {
         for(int i=0;i<pos-1;i++)
    {
        q=p;
        p=p->next;
    }
    q->next=p->next;
    printf("Removed player : %s\n",p->name);
    free(p);
    }
}
```



```
1.Create Injured list
2.Insert Player
3.Display Injured list
4.Delete Player
5.Exit
Enter choice : 2
Enter name : p3
Enter injured part : head
1.Create Injured list
2.Insert Player
3.Display Injured list
4.Delete Player
5.Exit
Enter choice : 3
Player : p1 | Injury : arm
Player : p2 | Injury : leg
Player : p3 | Injury : head

1.Create Injured list
2.Insert Player
3.Display Injured list
4.Delete Player
5.Exit
Enter choice : 4
Enter position : 1
Removed player : p1
1.Create Injured list
2.Insert Player
3.Display Injured list
4.Delete Player
5.Exit
Enter choice : 3
Player : p2 | Injury : leg
Player : p3 | Injury : head

1.Create Injured list
2.Insert Player
3.Display Injured list
4.Delete Player
5.Exit
Enter choice :
```

```
/*Manage bookings for sports facilities using a linked list.Operations:
Create a booking list.
Insert a new booking.
Delete a canceled or completed booking.
Display all current bookings.*/
#include<stdlib.h>
#include<stdio.h>
#include<string.h>
struct Sport
{
    char class[20];
    char facility[20];
    struct Sport *next;
};
struct Sport *first=NULL;
void create();
void insert();
void delete();
void display();
void main()
{
    int choice;
    do
    {
     printf("1.Create booking list  \n");
        printf("2.Insert booking\n");
        printf("3.Display booking list\n");
        printf("4.Delete booking\n");
        printf("5.Exit\n");
        printf("Enter choice : ");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
            create();
            break;
            case 2:
            insert();
            break;
            case 3:
```

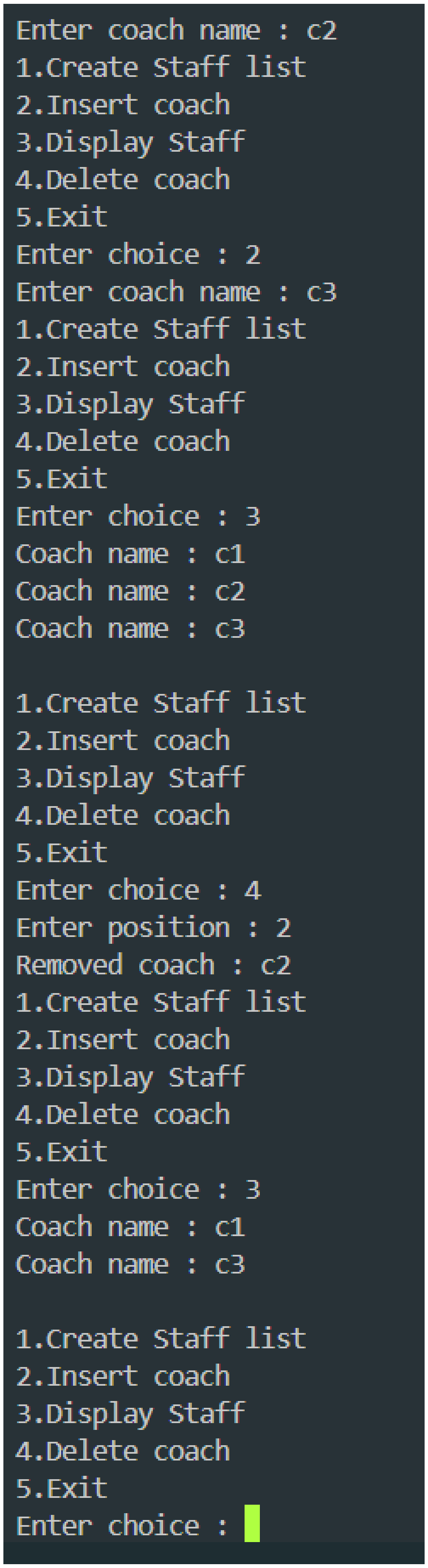
```
            display();
            break;
            case 4:
            delete();
            break;
            case 5:printf("Exiing..\n");
            break;
        }
    } while (choice!=5);

}
void create ()
{
    int n;
    struct Sport *ptr;
    ptr=first;
    printf("Enter size of booking list: ");
    scanf("%d",&n);
    for(int i=0;i<n;i++)
    {
        struct Sport* newitem=(struct Sport *)malloc(sizeof(struct
Sport));
        newitem->next=NULL;
        strcpy(newitem->class,"\0");
        strcpy(newitem->facility,"\0");
        if(first==NULL)
        {
            first=newitem;
            ptr=first;
        }
        else
        {
            ptr->next=newitem;
            ptr=ptr->next;
        }
    }
}
void insert()
{
    struct Sport *ptr;
    char name[20],f[20];
    printf("Enter class name : ");
    scanf("%s",name);
   printf("Enter facility name : ");
    scanf("%s",f);
    ptr=first;
    while(ptr!=0)
    {
        if(strcmp(ptr->class,"\0")==0)
        {
            strcpy(ptr->class,name);
            strcpy(ptr->facility,f);
            return;
        }
        ptr=ptr->next;
    }
}
void display()
{
    struct Sport *ptr;
    ptr=first;
    while(ptr!=NULL)
    {

        if(strcmp(ptr->class,"\0")!=0)
        {
            printf("Class : %s | Facility :
%s\n",ptr->class,ptr->facility);
        }

        ptr=ptr->next;
    }
    printf("\n");
}
void delete()
{
    int pos;
    struct Sport*p,*q;
    p=first;
    printf("Enter position : ");
    scanf("%d",&pos);
    if(pos==1)
    {
        first=first->next;
        printf("Removed Booking : %s\n",p->class);
        free(p);
    }
    else
    {
         for(int i=0;i<pos-1;i++)
    {
        q=p;
        p=p->next;
    }
    q->next=p->next;
    printf("Removed Booking : %s\n",p->class);
    free(p);
    }
}
```



```
2.Insert booking
3.Display booking list
4.Delete booking
5.Exit
Enter choice : 2
Enter class name :
c
Enter facility name : ground
1.Create booking list
2.Insert booking
3.Display booking list
4.Delete booking
5.Exit
Enter choice : 3
Class : A | Facility : gym
Class : B | Facility : hall
Class : c | Facility : ground

1.Create booking list
2.Insert booking
3.Display booking list
4.Delete booking
5.Exit
Enter choice : 4
Enter position : 3
Removed Booking : c
1.Create booking list
2.Insert booking
3.Display booking list
4.Delete booking
5.Exit
Enter choice : 3
Class : A | Facility : gym
Class : B | Facility : hall

1.Create booking list
2.Insert booking
3.Display booking list
4.Delete booking
5.Exit
Enter choice : 
```

```
/*Use a linked list to manage the coaching staff of a sports
team.Operations:
Create a coaching staff list.
Insert a new coach.
Delete a coach who leaves the team.
Display the current coaching staff.*/
#include<stdlib.h>
#include<stdio.h>
#include<string.h>
struct Staff
{
    char name[20];
    struct Staff *next;
};
struct Staff *first=NULL;
void create();
void insert();
void delete();
void display();
void main()
{
    int choice;
    do
    {
     printf("1.Create Staff list  \n");
        printf("2.Insert coach\n");
        printf("3.Display Staff\n");
        printf("4.Delete coach\n");
        printf("5.Exit\n");
        printf("Enter choice : ");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
            create();
            break;
            case 2:
            insert();
            break;
            case 3:
```

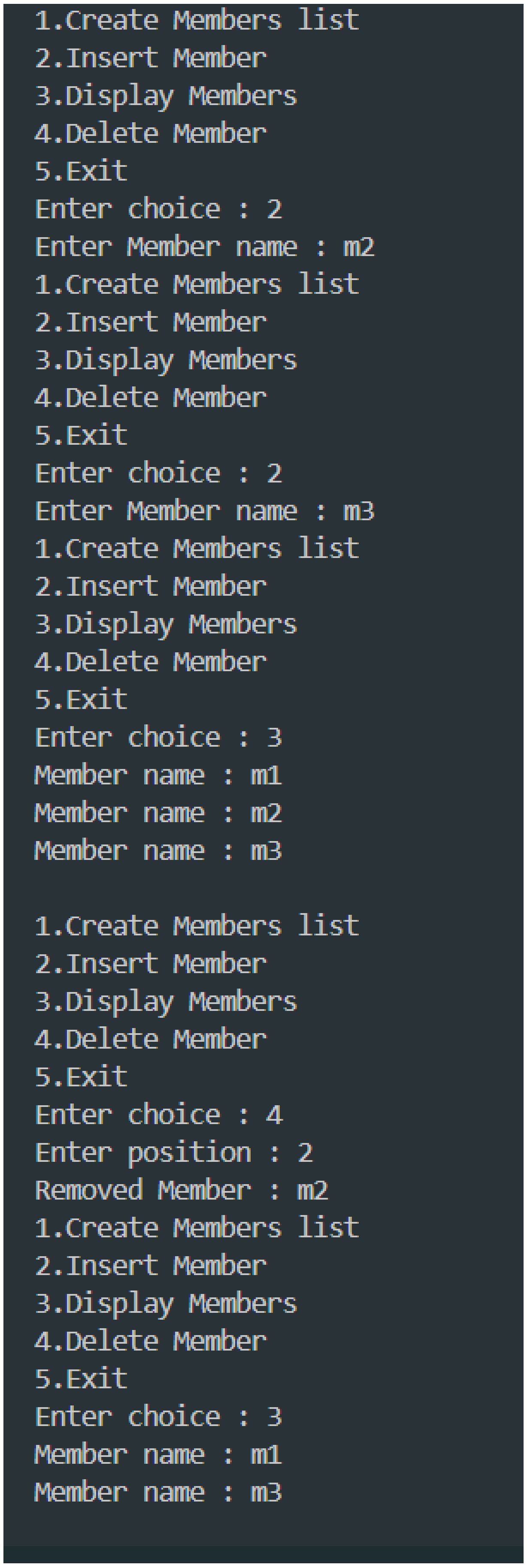
```
            display();
            break;
            case 4:
            delete();
            break;
            case 5:printf("Exiing..\n");
            break;
        }
    } while (choice!=5);

}
void create ()
{
    int n;
    struct Staff *ptr;
    ptr=first;
    printf("Enter size of Staff list : ");
    scanf("%d",&n);
    for(int i=0;i<n;i++)
    {
        struct Staff* newitem=(struct Staff *)malloc(sizeof(struct
Staff));
        newitem->next=NULL;
        strcpy(newitem->name,"\0");
        if(first==NULL)
        {
            first=newitem;
            ptr=first;
        }
        else
        {
            ptr->next=newitem;
            ptr=ptr->next;
        }
    }
}
void insert()
{
    struct Staff *ptr;
    char name[10];
    printf("Enter coach name : ");
    scanf("%s",name);
    ptr=first;
    while(ptr!=0)
    {
        if(strcmp(ptr->name,"\0")==0)
        {
            strcpy(ptr->name,name);
            return;
        }
        ptr=ptr->next;
    }
}
void display()
{
    struct Staff *ptr;
    ptr=first;
    while(ptr!=NULL)
    {

        if(strcmp(ptr->name,"\0")!=0)
        printf("Coach name : %s\n",ptr->name);
        ptr=ptr->next;
    }
    printf("\n");
}
void delete()
{
    int pos;
    struct Staff*p,*q;
    p=first;
    printf("Enter position : ");
    scanf("%d",&pos);
    if(pos==1)
    {
        first=first->next;
        printf("Removed coach : %s\n",p->name);
        free(p);
    }
    else
    {
         for(int i=0;i<pos-1;i++)
    {
        q=p;
        p=p->next;
    }
    q->next=p->next;
    printf("Removed coach : %s\n",p->name);
    free(p);
    }
}
```



```
Enter coach name : c2
1.Create Staff list
2.Insert coach
3.Display Staff
4.Delete coach
5.Exit
Enter choice : 2
Enter coach name : c3
1.Create Staff list
2.Insert coach
3.Display Staff
4.Delete coach
5.Exit
Enter choice : 3
Coach name : c1
Coach name : c2
Coach name : c3

1.Create Staff list
2.Insert coach
3.Display Staff
4.Delete coach
5.Exit
Enter choice : 4
Enter position : 2
Removed coach : c2
1.Create Staff list
2.Insert coach
3.Display Staff
4.Delete coach
5.Exit
Enter choice : 3
Coach name : c1
Coach name : c3

1.Create Staff list
2.Insert coach
3.Display Staff
4.Delete coach
5.Exit
Enter choice : 
```

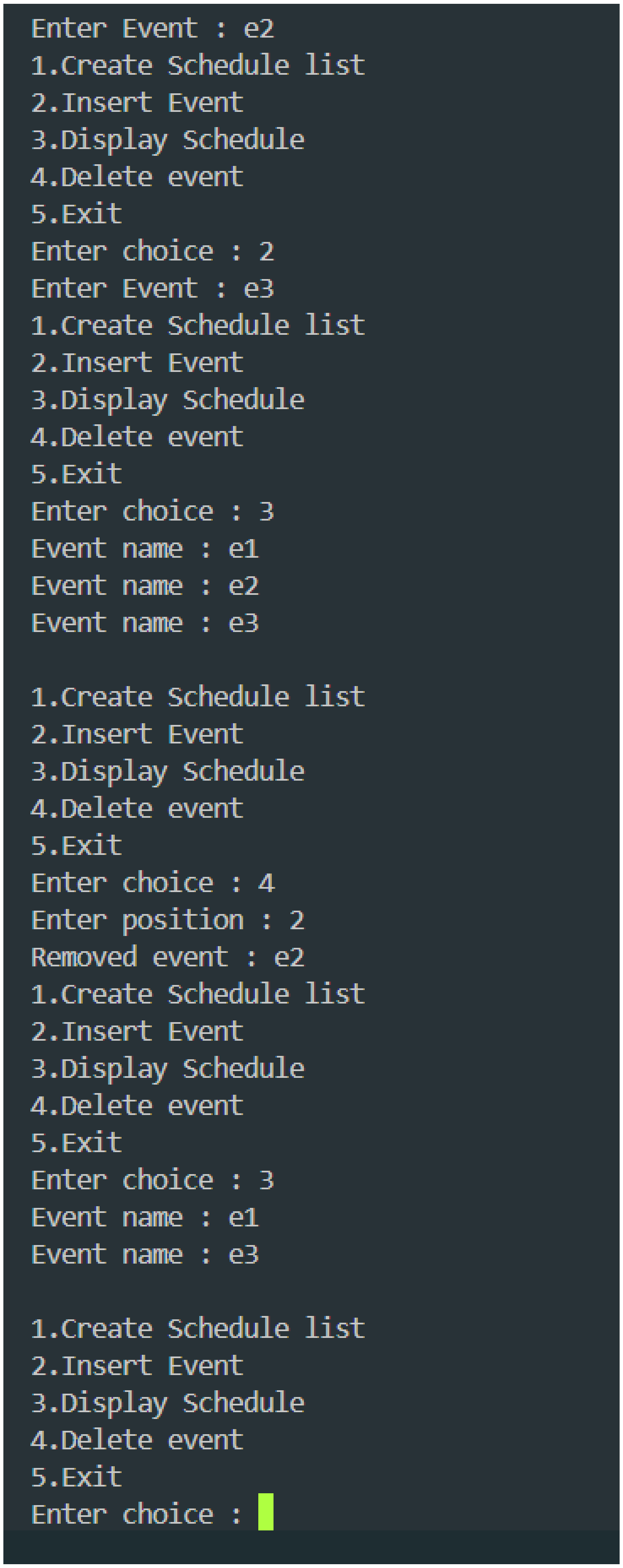
```c
/*Implement a linked list to manage memberships in a sports team’s fan
club.Operations:
Create a membership list.
Insert a new member.
Delete a member who cancels their membership.
Display all current members.*/
#include<stdlib.h>
#include<stdio.h>
#include<string.h>
struct Members
{
    char name[20];
    struct Members *next;
};
struct Members *first=NULL;
void create();
void insert();
void delete();
void display();
void main()
{
    int choice;
    do
    {
     printf("1.Create Members list  \n");
        printf("2.Insert Member\n");
        printf("3.Display Members\n");
        printf("4.Delete Member\n");
        printf("5.Exit\n");
        printf("Enter choice : ");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
            create();
            break;
            case 2:
            insert();
            break;
            case 3:
```

```
            display();
            break;
            case 4:
            delete();
            break;
            case 5:printf("Exiing..\n");
            break;
        }
    } while (choice!=5);

}
void create ()
{
    int n;
    struct Members *ptr;
    ptr=first;
    printf("Enter size of Members list : ");
    scanf("%d",&n);
    for(int i=0;i<n;i++)
    {
        struct Members* newitem=(struct Members *)malloc(sizeof(struct
Members));
        newitem->next=NULL;
        strcpy(newitem->name,"\0");
        if(first==NULL)
        {
            first=newitem;
            ptr=first;
        }
        else
        {
            ptr->next=newitem;
            ptr=ptr->next;
        }
    }
}
void insert()
{
    struct Members *ptr;
    char name[10];
    printf("Enter Member name : ");
    scanf("%s",name);
    ptr=first;
    while(ptr!=0)
    {
        if(strcmp(ptr->name,"\0")==0)
        {
            strcpy(ptr->name,name);
            return;
        }
        ptr=ptr->next;
    }
}
void display()
{
    struct Members *ptr;
    ptr=first;
    while(ptr!=NULL)
    {

        if(strcmp(ptr->name,"\0")!=0)
        printf("Member name : %s\n",ptr->name);
        ptr=ptr->next;
    }
    printf("\n");
}
void delete()
{
    int pos;
    struct Members*p,*q;
    p=first;
    printf("Enter position : ");
    scanf("%d",&pos);
    if(pos==1)
    {
        first=first->next;
        printf("Removed Member : %s\n",p->name);
        free(p);
    }
    else
    {
         for(int i=0;i<pos-1;i++)
    {
        q=p;
        p=p->next;
    }
    q->next=p->next;
    printf("Removed Member : %s\n",p->name);
    free(p);
    }
}
```



```
1.Create Members list
2.Insert Member
3.Display Members
4.Delete Member
5.Exit
Enter choice : 2
Enter Member name : m2
1.Create Members list
2.Insert Member
3.Display Members
4.Delete Member
5.Exit
Enter choice : 2
Enter Member name : m3
1.Create Members list
2.Insert Member
3.Display Members
4.Delete Member
5.Exit
Enter choice : 3
Member name : m1
Member name : m2
Member name : m3

1.Create Members list
2.Insert Member
3.Display Members
4.Delete Member
5.Exit
Enter choice : 4
Enter position : 2
Removed Member : m2
1.Create Members list
2.Insert Member
3.Display Members
4.Delete Member
5.Exit
Enter choice : 3
Member name : m1
Member name : m3
```

```
/*Use a linked list to manage the schedule of sports events.Operations:
Create an event schedule.
Insert a new event.
Delete a completed or canceled event.
Display the current event schedule*/
#include<stdlib.h>
#include<stdio.h>
#include<string.h>
struct Schedule
{
    char event[20];
    struct Schedule *next;
};
struct Schedule *first=NULL;
void create();
void insert();
void delete();
void display();
void main()
{
    int choice;
    do
    {
     printf("1.Create Schedule list  \n");
        printf("2.Insert Event\n");
        printf("3.Display Schedule\n");
        printf("4.Delete event\n");
        printf("5.Exit\n");
        printf("Enter choice : ");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
            create();
            break;
            case 2:
            insert();
            break;
            case 3:
            display();
            break;
            case 4:
            delete();
            break;
            case 5:printf("Exiing..\n");
            break;
        }
    } while (choice!=5);

}
void create ()
{
    int n;
    struct Schedule *ptr;
    ptr=first;
    printf("Enter size of Schedule list : ");
    scanf("%d",&n);
    for(int i=0;i<n;i++)
    {
        struct Schedule* newitem=(struct Schedule *)malloc(sizeof(struct
Schedule));
        newitem->next=NULL;
        strcpy(newitem->event,"\0");
        if(first==NULL)
        {
            first=newitem;
            ptr=first;
        }
        else
        {
            ptr->next=newitem;
            ptr=ptr->next;
        }
    }
}
void insert()
{
    struct Schedule *ptr;
    char name[10];
    printf("Enter Event : ");
    scanf("%s",name);
    ptr=first;
    while(ptr!=0)
    {
        if(strcmp(ptr->event,"\0")==0)
        {
            strcpy(ptr->event,name);
            return;
        }
        ptr=ptr->next;
    }
}
void display()
{
    struct Schedule *ptr;
    ptr=first;
    while(ptr!=NULL)
    {

        if(strcmp(ptr->event,"\0")!=0)
        printf("Event name : %s\n",ptr->event);
        ptr=ptr->next;
    }
    printf("\n");
}
void delete()
{
    int pos;
    struct Schedule*p,*q;
    p=first;
    printf("Enter position : ");
    scanf("%d",&pos);
    if(pos==1)
    {
        first=first->next;
        printf("Removed event : %s\n",p->event);
        free(p);
    }
    else
    {
         for(int i=0;i<pos-1;i++)
    {
        q=p;
        p=p->next;
    }
    q->next=p->next;
    printf("Removed event : %s\n",p->event);
    free(p);
    }
}
```

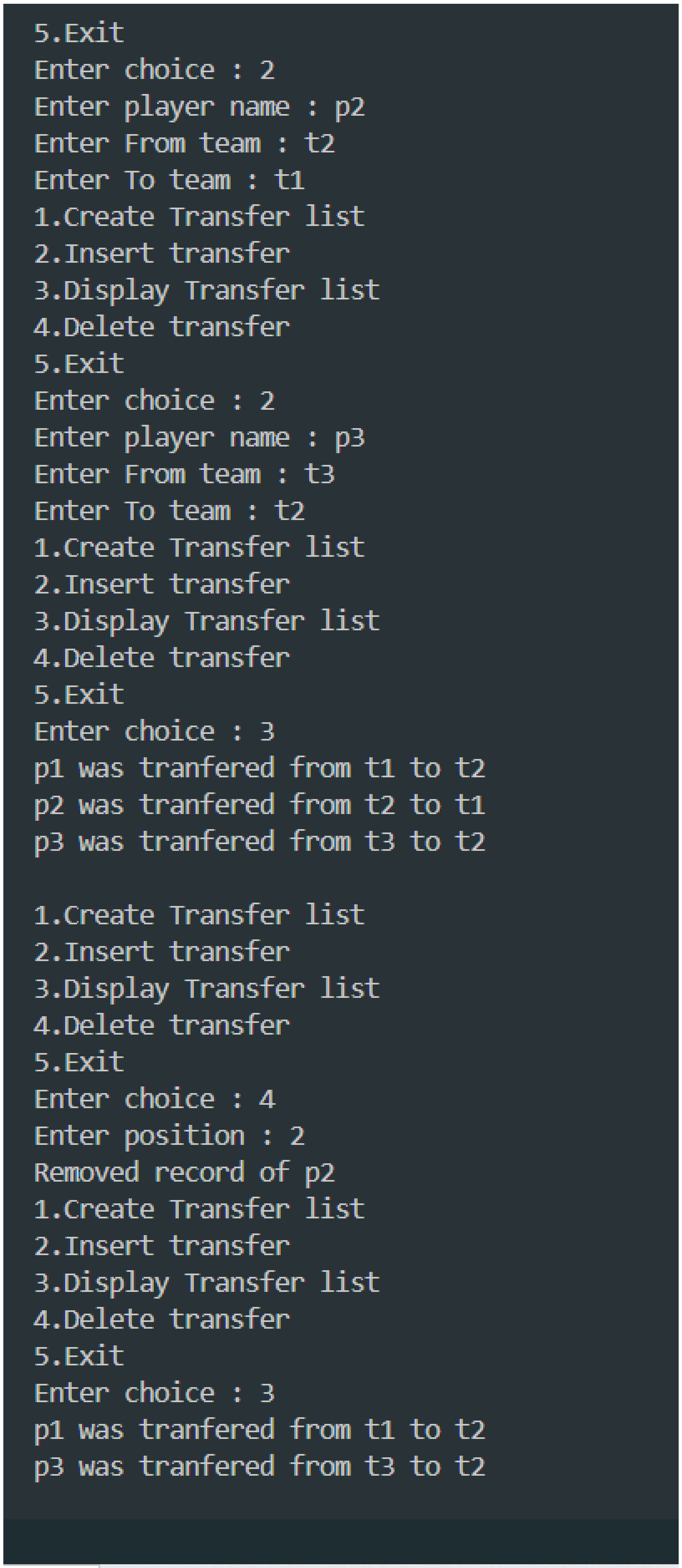


```
Enter Event : e2
1.Create Schedule list
2.Insert Event
3.Display Schedule
4.Delete event
5.Exit
Enter choice : 2
Enter Event : e3
1.Create Schedule list
2.Insert Event
3.Display Schedule
4.Delete event
5.Exit
Enter choice : 3
Event name : e1
Event name : e2
Event name : e3

1.Create Schedule list
2.Insert Event
3.Display Schedule
4.Delete event
5.Exit
Enter choice : 4
Enter position : 2
Removed event : e2
1.Create Schedule list
2.Insert Event
3.Display Schedule
4.Delete event
5.Exit
Enter choice : 3
Event name : e1
Event name : e3

1.Create Schedule list
2.Insert Event
3.Display Schedule
4.Delete event
5.Exit
Enter choice : 
```

```
/*Maintain a linked list to track player transfers between
teams.Operations:
Create a transfer record list.
Insert a new transfer record.
Delete an outdated or erroneous transfer record.
Display all current transfer records.*/
#include<stdlib.h>
#include<stdio.h>
#include<string.h>
struct Transfer
{
    char name[20];
    char team1[20];
    char team2[20];
    struct Transfer *next;
};
struct Transfer *first=NULL;
void create();
void insert();
void delete();
void display();
void main()
{
    int choice;
    do
    {
     printf("1.Create Transfer list  \n");
        printf("2.Insert transfer\n");
        printf("3.Display Transfer list\n");
        printf("4.Delete transfer\n");
        printf("5.Exit\n");
        printf("Enter choice : ");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
            create();
            break;
            case 2:
            insert();
            break;
            case 3:
            display();
            break;
            case 4:
            delete();
            break;
            case 5:printf("Exiing..\n");
            break;
        }
    } while (choice!=5);

}
void create ()
{
    int n;
    struct Transfer *ptr;
    ptr=first;
    printf("Enter size of Transfer list : ");
    scanf("%d",&n);
    for(int i=0;i<n;i++)
    {
        struct Transfer* newitem=(struct Transfer *)malloc(sizeof(struct
Transfer));
        newitem->next=NULL;
        strcpy(newitem->team1,"\0");
        strcpy(newitem->team2,"\0");
        strcpy(newitem->name,"\0");
        if(first==NULL)
        {
            first=newitem;
            ptr=first;
        }
        else
        {
            ptr->next=newitem;
            ptr=ptr->next;
        }
    }
}
```

```
void insert()
{
    struct Transfer *ptr;
    char name1[20],name2[20],name[20];
    printf("Enter player name : ");
    scanf("%s",name);
    printf("Enter From team : ");
    scanf("%s",name1);
    printf("Enter To team : ");
    scanf("%s",name2);
    ptr=first;
    while(ptr!=0)
    {
        if(strcmp(ptr->name,"\0")==0)
        {
            strcpy(ptr->name,name);
            strcpy(ptr->team1,name1);
            strcpy(ptr->team2,name2);
            return;
        }
        ptr=ptr->next;
    }
}
void display()
{
    struct Transfer *ptr;
    ptr=first;
    while(ptr!=NULL)
    {

        if(strcmp(ptr->name,"\0")!=0)
        {
            printf("%s was tranfered from %s to
%s\n",ptr->name,ptr->team1,ptr->team2);
        }

        ptr=ptr->next;
    }
    printf("\n");
}
```

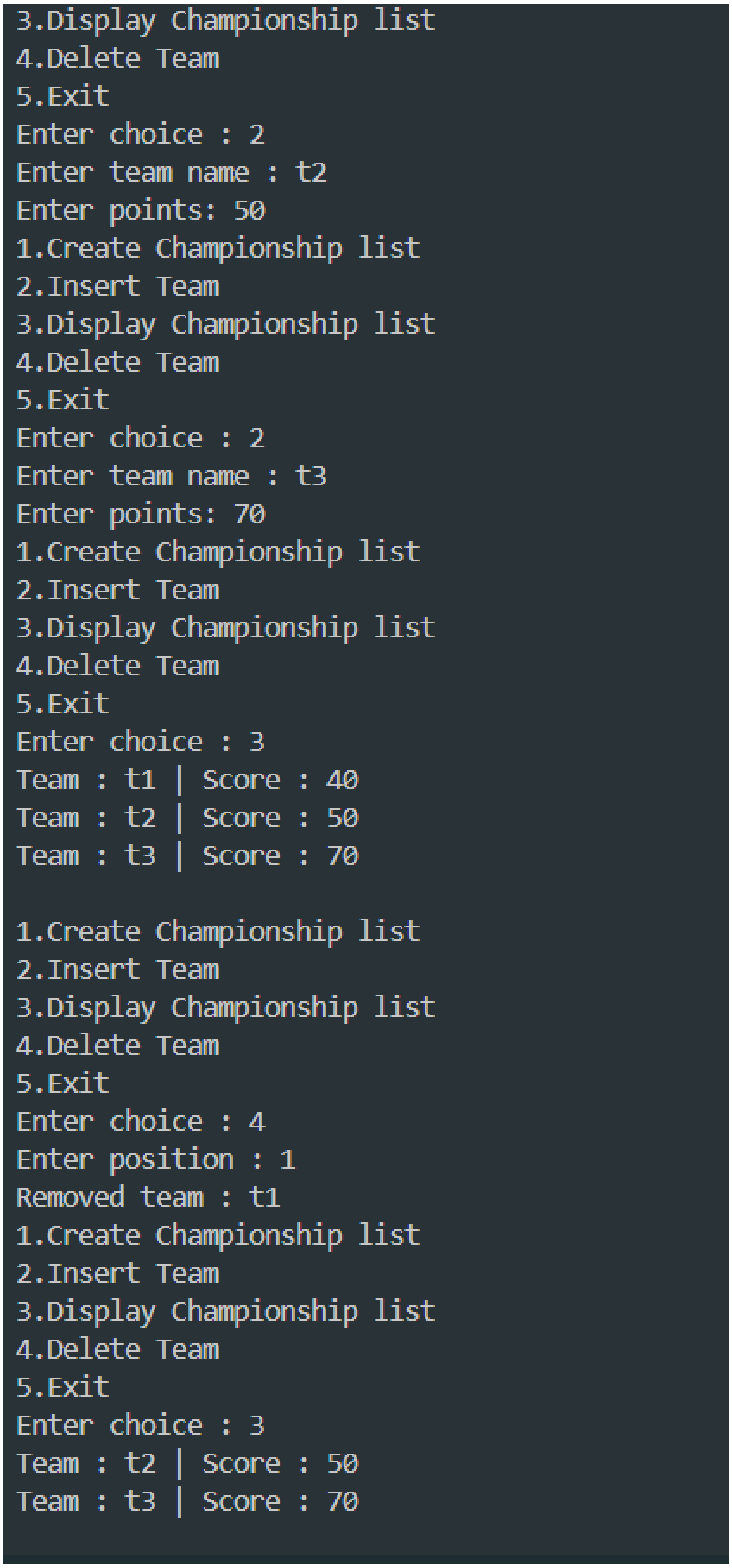
```c
void delete()
{
    int pos;
    struct Transfer*p,*q;
    p=first;
    printf("Enter position : ");
    scanf("%d",&pos);
    if(pos==1)
    {
        first=first->next;
        printf("Removed record of %s\n",p->name);
        free(p);
    }
    else
    {
         for(int i=0;i<pos-1;i++)
    {
        q=p;
        p=p->next;
    }
    q->next=p->next;
    printf("Removed record of %s\n",p->name);
    free(p);
    }
}
```



```
5.Exit
Enter choice : 2
Enter player name : p2
Enter From team : t2
Enter To team : t1
1.Create Transfer list
2.Insert transfer
3.Display Transfer list
4.Delete transfer
5.Exit
Enter choice : 2
Enter player name : p3
Enter From team : t3
Enter To team : t2
1.Create Transfer list
2.Insert transfer
3.Display Transfer list
4.Delete transfer
5.Exit
Enter choice : 3
p1 was tranfered from t1 to t2
p2 was tranfered from t2 to t1
p3 was tranfered from t3 to t2

1.Create Transfer list
2.Insert transfer
3.Display Transfer list
4.Delete transfer
5.Exit
Enter choice : 4
Enter position : 2
Removed record of p2
1.Create Transfer list
2.Insert transfer
3.Display Transfer list
4.Delete transfer
5.Exit
Enter choice : 3
p1 was tranfered from t1 to t2
p3 was tranfered from t3 to t2
```

```c
/*: Implement a linked list to track championship points for
teams.Operations:
Create a points tracker list.
Insert a new points entry.
Delete an incorrect or outdated points entry.
Display all current points standings.*/
#include<stdlib.h>
#include<stdio.h>
#include<string.h>
struct Championship
{
    char name[20];
    int points;
    struct Championship *next;
};
struct Championship *first=NULL;
void create();
void insert();
void delete();
void display();
void main()
{
    int choice;
    do
    {
     printf("1.Create Championship list  \n");
        printf("2.Insert Team\n");
        printf("3.Display Championship list\n");
        printf("4.Delete Team\n");
        printf("5.Exit\n");
        printf("Enter choice : ");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
            create();
            break;
            case 2:
            insert();
            break;
            case 3:
            display();
            break;
            case 4:
            delete();
            break;
            case 5:printf("Exiing..\n");
            break;
        }
    } while (choice!=5);

}
void create ()
{
    int n;
    struct Championship *ptr;
    ptr=first;
    printf("Enter size of Championship list : ");
    scanf("%d",&n);
    for(int i=0;i<n;i++)
    {
        struct Championship* newitem=(struct Championship
*)malloc(sizeof(struct Championship));
        newitem->next=NULL;
        strcpy(newitem->name,"\0");
        newitem->points=-1;
        if(first==NULL)
        {
            first=newitem;
            ptr=first;
        }
        else
        {
            ptr->next=newitem;
            ptr=ptr->next;
        }
    }
}
void insert()
{
```

```c
    struct Championship *ptr;
    char name[20];
    int n;
    printf("Enter team name : ");
    scanf("%s",name);
    printf("Enter points: ");
    scanf("%d",&n);
    ptr=first;
    while(ptr!=0)
    {
        if(strcmp(ptr->name,"\0")==0)
        {
            strcpy(ptr->name,name);
            ptr->points=n;
            return;
        }
        ptr=ptr->next;
    }
}
void display()
{
    struct Championship *ptr;
    ptr=first;
    while(ptr!=NULL)
    {

        if(strcmp(ptr->name,"\0")!=0)
        {
            printf("Team : %s | Score : %d\n",ptr->name,ptr->points);
        }

        ptr=ptr->next;
    }
    printf("\n");
}
void delete()
{
    int pos;
    struct Championship*p,*q;
    p=first;
    printf("Enter position : ");
    scanf("%d",&pos);
    if(pos==1)
    {
        first=first->next;
        printf("Removed team : %s\n",p->name);
        free(p);
    }
    else
    {
         for(int i=0;i<pos-1;i++)
    {
        q=p;
        p=p->next;
    }
    q->next=p->next;
    printf("Removed team : %s\n",p->name);
    free(p);
    }
}
```



```
3.Display Championship list
4.Delete Team
5.Exit
Enter choice : 2
Enter team name : t2
Enter points: 50
1.Create Championship list
2.Insert Team
3.Display Championship list
4.Delete Team
5.Exit
Enter choice : 2
Enter team name : t3
Enter points: 70
1.Create Championship list
2.Insert Team
3.Display Championship list
4.Delete Team
5.Exit
Enter choice : 3
Team : t1 | Score : 40
Team : t2 | Score : 50
Team : t3 | Score : 70

1.Create Championship list
2.Insert Team
3.Display Championship list
4.Delete Team
5.Exit
Enter choice : 4
Enter position : 1
Removed team : t1
1.Create Championship list
2.Insert Team
3.Display Championship list
4.Delete Team
5.Exit
Enter choice : 3
Team : t2 | Score : 50
Team : t3 | Score : 70
```