```c
/*: Implement a stack-based system using arrays to record the sequence of
flight paths an aircraft takes.
Use a switch-case menu with options:
1: Add a new path (push)
2: Undo the last path (pop)
3: Display the current flight path stack
4: Peek at the top path
5: Search for a specific path
6: Exit*/
#include<stdio.h>
#include<stdlib.h>
struct Stack
{
int size;
int top;
int *p;
};
void create(struct Stack *);
void push(struct Stack *);
void pop(struct Stack *);
void display(struct Stack);
void top(struct Stack);
int peek(struct Stack);
void main()
{
    struct Stack st;
    create(&st);
    int choice,f;
    do
    {
       printf("1: Add a new path (push)\n");
       printf("2: Undo the last path (pop)\n");
       printf("3: Display the current flight path stack\n");
       printf("4: Peek at the top path\n");
       printf("5: Search for a specific path\n");
       printf("6: Exit\n");
       printf("Enter choice : ");
       scanf("%d",&choice);
       switch(choice)
       {
```

```
        case 1:push(&st);
        break;
        case 2:pop(&st);
        break;
        case 3:display(st);
        break;
        case 4:top(st);
        break;
        case 5:f=peek(st);
        if(f==0)
        printf("No path found\n");
        else
        printf("Path found\n");
        break;
        case 6:printf("Exiting.....\n");
        break;
        default :printf("Enter valid option\n");
        break;
       }
    } while (choice!=6);

}
void create(struct Stack *st){
    printf("Enter The Size : ");
    scanf("%d",&st->size);
    st->top = -1;
    st->p = (int *)malloc((st->size) * sizeof(int));
}
void push(struct Stack *st)
{
    int x;
    printf("Enter path no : ");
    scanf("%d",&x);
    if(st->top==st->size-1)
    printf("Stack full\n");
    else
    {
       st->top++;
       st->p[st->top]=x;
    }
```

```
}
void pop(struct Stack *st)
{
    int x;
    if(st->top==-1)
    printf("Stack is empty\n");
    else
    {
        x=st->p[st->top];
        st->top--;
    }
    printf("Path no:%d has been removed\n");
}
void display(struct Stack st)
{
    for(int i=st.top;i>=0;i--)
    {
        printf("%d\n",st.p[i]);
    }
}
void  top(struct Stack st)
{
    if(st.top==-1)
    printf("Stack is empty\n");
    else
    printf("Top path is %d\n",st.p[st.top]);
}
int peek(struct Stack st)
{
    int x;
    printf("Enter the path no to search :");
    scanf("%d",&x);
    if(st.top==-1)
    printf("Stack is empty\n");
    else
    {
        for(int i=st.top;i>=0;i--)
        {
            if(st.p[i]==x)
            return 1;
```

```
        }
    }
    return 0;
}
/*Develop a stack using arrays to manage the sequence of satellite
deployments from a spacecraft. Include a switch-case menu with options:
1: Push a new satellite deployment
2: Pop the last deployment
3: View the deployment sequence
4: Peek at the latest deployment
5: Search for a specific deployment
6: Exit*/
#include<stdio.h>
#include<stdlib.h>
struct Stack
{
int size;
int top;
int *p;
};
void create(struct Stack *);
void push(struct Stack *);
void pop(struct Stack *);
void display(struct Stack);
void top(struct Stack);
int peek(struct Stack);
void main()
{
    struct Stack st;
    create(&st);
    int choice,f;
    do
    {
       printf("1: Push a new satellite deployment\n");
       printf("2: Pop the last deployment\n");
       printf("3: View the deployment sequence\n");
       printf("4: Peek at the latest deployment\n");
       printf("5: Search for a specific deployment\n");
       printf("6: Exit\n");
       printf("Enter choice : ");
```

```
        scanf("%d",&choice);
        switch(choice)
        {
         case 1:push(&st);
         break;
         case 2:pop(&st);
         break;
         case 3:display(st);
         break;
         case 4:top(st);
         break;
         case 5:f=peek(st);
         if(f==0)
         printf("No satelite found\n");
         else
         printf("Satelite found\n");
         break;
         case 6:printf("Exiting.....\n");
         break;
         default :printf("Enter valid option\n");
         break;
        }
    } while (choice!=6);

}
void create(struct Stack *st){
    printf("Enter The Size : ");
    scanf("%d",&st->size);
    st->top = -1;
    st->p = (int *)malloc((st->size) * sizeof(int));
}
void push(struct Stack *st)
{
    int x;
    printf("Enter satelite no : ");
    scanf("%d",&x);
    if(st->top==st->size-1)
    printf("Stack full\n");
    else
    {
```

```
        st->top++;
        st->p[st->top]=x;
    }
}
void pop(struct Stack *st)
{
    int x;
    if(st->top==-1)
    printf("Stack is empty\n");
    else
    {
        x=st->p[st->top];
        st->top--;
    }
    printf("Satelite no:%d has been removed\n");
}
void display(struct Stack st)
{
    for(int i=st.top;i>=0;i--)
    {
        printf("%d\n",st.p[i]);
    }
}
void  top(struct Stack st)
{
    if(st.top==-1)
    printf("Stack is empty\n");
    else
    printf("Top Satelite is %d\n",st.p[st.top]);
}
int peek(struct Stack st)
{
    int x;
    printf("Enter the satelite no to search :");
    scanf("%d",&x);
    if(st.top==-1)
    printf("Stack is empty\n");
    else
    {
        for(int i=st.top;i>=0;i--)
```

```
        {
            if(st.p[i]==x)
            return 1;
        }
    }
    return 0;
}
/*Create a stack for a rocket launch checklist using arrays. Implement a
switch-case menu with options:
1: Add a checklist item (push)
2: Remove the last item (pop)
3: Display the current checklist
4: Peek at the top checklist item
5: Search for a specific checklist item
6: Exit*/
#include<stdio.h>
#include<stdlib.h>
struct Stack
{
int size;
int top;
int *p;
};
void create(struct Stack *);
void push(struct Stack *);
void pop(struct Stack *);
void display(struct Stack);
void top(struct Stack);
int peek(struct Stack);
void main()
{
    struct Stack st;
    create(&st);
    int choice,f;
    do
    {
       printf("1: Add a checklist item\n");
       printf("2: Remove the last item\n");
       printf("3: Display the current checklist\n");
       printf("4: Peek at the top checklist item\n");
```

```
       printf("5: Search for a specific checklist item\n");
       printf("6: Exit\n");
       printf("Enter choice : ");
       scanf("%d",&choice);
       switch(choice)
       {
        case 1:push(&st);
        break;
        case 2:pop(&st);
        break;
        case 3:display(st);
        break;
        case 4:top(st);
        break;
        case 5:f=peek(st);
        if(f==0)
        printf("No item found\n");
        else
        printf("Item found\n");
        break;
        case 6:printf("Exiting.....\n");
        break;
        default :printf("Enter valid option\n");
        break;
       }
    } while (choice!=6);

}
void create(struct Stack *st){
    printf("Enter The Size : ");
    scanf("%d",&st->size);
    st->top = -1;
    st->p = (int *)malloc((st->size) * sizeof(int));
}
void push(struct Stack *st)
{
    int x;
    printf("Enter item : ");
    scanf("%d",&x);
    if(st->top==st->size-1)
```

```
    printf("Stack full\n");
    else
    {
       st->top++;
       st->p[st->top]=x;
    }
}
void pop(struct Stack *st)
{
    int x;
    if(st->top==-1)
    printf("Stack is empty\n");
    else
    {
        x=st->p[st->top];
        st->top--;
    }
    printf("Item no:%d has been removed\n");
}
void display(struct Stack st)
{
    for(int i=st.top;i>=0;i--)
    {
        printf("%d\n",st.p[i]);
    }
}
void  top(struct Stack st)
{
    if(st.top==-1)
    printf("Stack is empty\n");
    else
    printf("Top item is %d\n",st.p[st.top]);
}
int peek(struct Stack st)
{
    int x;
    printf("Enter the item no to search :");
    scanf("%d",&x);
    if(st.top==-1)
    printf("Stack is empty\n");
```

```
    else
    {
        for(int i=st.top;i>=0;i--)
        {
            if(st.p[i]==x)
            return 1;
        }
    }
    return 0;
}
/*Design a stack-based task manager for space missions using arrays.
Include a switch-case menu with options:
1: Add a task (push)
2: Mark the last task as completed (pop)
3: List all pending tasks
4: Peek at the most recent task
5: Search for a specific task
6: Exit*/
#include<stdio.h>
#include<stdlib.h>
struct Stack
{
int size;
int top;
int *p;
};
void create(struct Stack *);
void push(struct Stack *);
void pop(struct Stack *);
void display(struct Stack);
void top(struct Stack);
int peek(struct Stack);
void main()
{
    struct Stack st;
    create(&st);
    int choice,f;
    do
    {
       printf("1: Add a task (push)\n");
```

```
        printf("2: Mark the last task as completed (pop)\n");
        printf("3: List all pending taskst\n");
        printf("4: Peek at the most recent task\n");
        printf("5: Search for a specific checklist item\n");
        printf("6: Exit\n");
        printf("Enter choice : ");
        scanf("%d",&choice);
        switch(choice)
        {
         case 1:push(&st);
         break;
         case 2:pop(&st);
         break;
         case 3:display(st);
         break;
         case 4:top(st);
         break;
         case 5:f=peek(st);
         if(f==0)
         printf("No task found\n");
         else
         printf("Task found\n");
         break;
         case 6:printf("Exiting.....\n");
         break;
         default :printf("Enter valid option\n");
         break;
        }
    } while (choice!=6);

}
void create(struct Stack *st){
    printf("Enter The Size : ");
    scanf("%d",&st->size);
    st->top = -1;
    st->p = (int *)malloc((st->size) * sizeof(int));
}
void push(struct Stack *st)
{
    int x;
```

```
    printf("Enter task id : ");
    scanf("%d",&x);
    if(st->top==st->size-1)
    printf("Stack full\n");
    else
    {
       st->top++;
       st->p[st->top]=x;
    }
}
void pop(struct Stack *st)
{
    int x;
    if(st->top==-1)
    printf("Stack is empty\n");
    else
    {
        x=st->p[st->top];
        st->top--;
    }
    printf("Task id :%d has been removed\n");
}
void display(struct Stack st)
{
    for(int i=st.top;i>=0;i--)
    {
        printf("%d\n",st.p[i]);
    }
}
void  top(struct Stack st)
{
    if(st.top==-1)
    printf("Stack is empty\n");
    else
    printf("Top task is %d\n",st.p[st.top]);
}
int peek(struct Stack st)
{
    int x;
    printf("Enter the task id to search :");
```

```
    scanf("%d",&x);
    if(st.top==-1)
    printf("Stack is empty\n");
    else
    {
        for(int i=st.top;i>=0;i--)
        {
            if(st.p[i]==x)
            return 1;
        }
    }
    return 0;
}
/*Use a stack to manage the countdown sequence for a rocket launch.
Implement a switch-case menu with options:
1: Add a countdown step (push)
2: Remove the last step (pop)
3: Display the current countdown
4: Peek at the next countdown step
5: Search for a specific countdown step
6: Exit*/
#include<stdio.h>
#include<stdlib.h>
struct Stack
{
int size;
int top;
int *p;
};
void create(struct Stack *);
void push(struct Stack *);
void pop(struct Stack *);
void display(struct Stack);
void top(struct Stack);
int peek(struct Stack);
void main()
{
    struct Stack st;
    create(&st);
    int choice,f;
```

```c
    do
    {
       printf("1: Add a countdown step (push)\n");
       printf("2: Remove the last step (pop)\n");
       printf("3: Display the current countdown\n");
       printf("4: Peek at the next countdown step\n");
       printf("5: Search for a specific countdown step\n");
       printf("6: Exit\n");
       printf("Enter choice : ");
       scanf("%d",&choice);
       switch(choice)
       {
        case 1:push(&st);
        break;
        case 2:pop(&st);
        break;
        case 3:display(st);
        break;
        case 4:top(st);
        break;
        case 5:f=peek(st);
        if(f==0)
        printf("No step found\n");
        else
        printf("Step found\n");
        break;
        case 6:printf("Exiting.....\n");
        break;
        default :printf("Enter valid option\n");
        break;
       }
    } while (choice!=6);

}
void create(struct Stack *st){
    printf("Enter The Size : ");
    scanf("%d",&st->size);
    st->top = -1;
    st->p = (int *)malloc((st->size) * sizeof(int));
}
```

```c
void push(struct Stack *st)
{
    int x;
    printf("Enter Step no : ");
    scanf("%d",&x);
    if(st->top==st->size-1)
    printf("Stack full\n");
    else
    {
       st->top++;
       st->p[st->top]=x;
    }
}
void pop(struct Stack *st)
{
    int x;
    if(st->top==-1)
    printf("Stack is empty\n");
    else
    {
        x=st->p[st->top];
        st->top--;
    }
    printf("Step :%d has been removed\n");
}
void display(struct Stack st)
{
    for(int i=st.top;i>=0;i--)
    {
        printf("%d\n",st.p[i]);
    }
}
void  top(struct Stack st)
{
    if(st.top==-1)
    printf("Stack is empty\n");
    else
    printf("Top step is %d\n",st.p[st.top]);
}
int peek(struct Stack st)
```

```
{
    int x;
    printf("Enter the step no to search :");
    scanf("%d",&x);
    if(st.top==-1)
    printf("Stack is empty\n");
    else
    {
        for(int i=st.top;i>=0;i--)
        {
            if(st.p[i]==x)
            return 1;
        }
    }
    return 0;
}
/*Implement a stack to keep track of maintenance logs for an aircraft. Use
a switch-case menu with options:
1: Add a new log (push)
2: Remove the last log (pop)
3: View all maintenance logs
4: Peek at the latest maintenance log
5: Search for a specific maintenance log
6: Exit*/
#include<stdio.h>
#include<stdlib.h>
struct Stack
{
int size;
int top;
int *p;
};
void create(struct Stack *);
void push(struct Stack *);
void pop(struct Stack *);
void display(struct Stack);
void top(struct Stack);
int peek(struct Stack);
void main()
{
```

```c
    struct Stack st;
    create(&st);
    int choice,f;
    do
    {
       printf("1: Add a new log (push)\n");
       printf("2: Remove the last log (pop)\n");
       printf("3: View all maintenance logs\n");
       printf("4: Peek at the latest maintenance log\n");
       printf("5: Search for a specific maintenance log\n");
       printf("6: Exit\n");
       printf("Enter choice : ");
       scanf("%d",&choice);
       switch(choice)
       {
        case 1:push(&st);
        break;
        case 2:pop(&st);
        break;
        case 3:display(st);
        break;
        case 4:top(st);
        break;
        case 5:f=peek(st);
        if(f==0)
        printf("No log found\n");
        else
        printf("Log found\n");
        break;
        case 6:printf("Exiting.....\n");
        break;
        default :printf("Enter valid option\n");
        break;
       }
    } while (choice!=6);

}
void create(struct Stack *st){
    printf("Enter The Size : ");
    scanf("%d",&st->size);
```

```
    st->top = -1;
    st->p = (int *)malloc((st->size) * sizeof(int));
}
void push(struct Stack *st)
{
    int x;
    printf("Enter Log id : ");
    scanf("%d",&x);
    if(st->top==st->size-1)
    printf("Stack full\n");
    else
    {
       st->top++;
       st->p[st->top]=x;
    }
}
void pop(struct Stack *st)
{
    int x;
    if(st->top==-1)
    printf("Stack is empty\n");
    else
    {
        x=st->p[st->top];
        st->top--;
    }
    printf("Log id :%d has been removed\n");
}
void display(struct Stack st)
{
    for(int i=st.top;i>=0;i--)
    {
        printf("%d\n",st.p[i]);
    }
}
void  top(struct Stack st)
{
    if(st.top==-1)
    printf("Stack is empty\n");
    else
```

```
    printf("Top step is %d\n",st.p[st.top]);
}
int peek(struct Stack st)
{
    int x;
    printf("Enter the log id to search :");
    scanf("%d",&x);
    if(st.top==-1)
    printf("Stack is empty\n");
    else
    {
        for(int i=st.top;i>=0;i--)
        {
            if(st.p[i]==x)
            return 1;
        }
    }
    return 0;
}
/*Develop a stack for the sequence of steps in a spacecraft docking
procedure. Implement a switch-case menu with options:
1: Push a new step
2: Pop the last step
3: Display the procedure steps
4: Peek at the next step in the procedure
5: Search for a specific step
6: Exit*/
#include<stdio.h>
#include<stdlib.h>
struct Stack
{
int size;
int top;
int *p;
};
void create(struct Stack *);
void push(struct Stack *);
void pop(struct Stack *);
void display(struct Stack);
void top(struct Stack);
```

```c
int peek(struct Stack);
void main()
{
    struct Stack st;
    create(&st);
    int choice,f;
    do
    {
       printf("1: Push a new step\n");
       printf("2: Pop the last step\n");
       printf("3: Display the procedure steps\n");
       printf("4: Peek at the next step in the procedure\n");
       printf("5: Search for a specific step\n");
       printf("6: Exit\n");
       printf("Enter choice : ");
       scanf("%d",&choice);
       switch(choice)
       {
        case 1:push(&st);
        break;
        case 2:pop(&st);
        break;
        case 3:display(st);
        break;
        case 4:top(st);
        break;
        case 5:f=peek(st);
        if(f==0)
        printf("No step found\n");
        else
        printf("Step found\n");
        break;
        case 6:printf("Exiting.....\n");
        break;
        default :printf("Enter valid option\n");
        break;
       }
    } while (choice!=6);

}
```

```c
void create(struct Stack *st){
    printf("Enter The Size : ");
    scanf("%d",&st->size);
    st->top = -1;
    st->p = (int *)malloc((st->size) * sizeof(int));
}
void push(struct Stack *st)
{
    int x;
    printf("Enter Step no : ");
    scanf("%d",&x);
    if(st->top==st->size-1)
    printf("Stack full\n");
    else
    {
       st->top++;
       st->p[st->top]=x;
    }
}
void pop(struct Stack *st)
{
    int x;
    if(st->top==-1)
    printf("Stack is empty\n");
    else
    {
        x=st->p[st->top];
        st->top--;
    }
    printf("Step no :%d has been removed\n");
}
void display(struct Stack st)
{
    for(int i=st.top;i>=0;i--)
    {
        printf("%d\n",st.p[i]);
    }
}
void  top(struct Stack st)
{
```

```c
    if(st.top==-1)
    printf("Stack is empty\n");
    else
    printf("Top step is %d\n",st.p[st.top]);
}
int peek(struct Stack st)
{
    int x;
    printf("Enter the log id to search :");
    scanf("%d",&x);
    if(st.top==-1)
    printf("Stack is empty\n");
    else
    {
        for(int i=st.top;i>=0;i--)
        {
            if(st.p[i]==x)
            return 1;
        }
    }
    return 0;
}
/*Create a stack to record the command history sent from mission control.
Use a switch-case menu with options:
1: Add a command (push)
2: Undo the last command (pop)
3: View the command history
4: Peek at the most recent command
5: Search for a specific command
6: Exit*/
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
struct Stack
{
int size;
int top;
char **c;
};
void create(struct Stack *);
```

```
void push(struct Stack *);
void pop(struct Stack *);
void display(struct Stack);
void top(struct Stack);
int peek(struct Stack);
void main()
{
    struct Stack st;
    create(&st);
    int choice,f;
    do
    {
       printf("1: Add a command (push)\n");
       printf("2: Undo the last command\n");
       printf("3: View the command history\n");
       printf("4:  Peek at the most recent command\n");
       printf("5: Search for a specific command\n");
       printf("6: Exit\n");
       printf("Enter choice : ");
       scanf("%d",&choice);
       switch(choice)
       {
        case 1:push(&st);
        break;
        case 2:pop(&st);
        break;
        case 3:display(st);
        break;
        case 4:top(st);
        break;
        case 5:f=peek(st);
        if(f==0)
        printf("No command found\n");
        else
        printf("Command found\n");
        break;
        case 6:printf("Exiting.....\n");
        break;
        default :printf("Enter valid option\n");
        break;
```

```
        }
    } while (choice!=6);

}
void create(struct Stack *st){
    printf("Enter The Size : ");
    scanf("%d",&st->size);
    st->top = -1;
    st->c = (char **)malloc((st->size) * sizeof(char*));
    for(int i=0;i<st->size;i++)
    st->c[i]=(char *)malloc(10*sizeof(char));
}
void push(struct Stack *st)
{
    char x[10];
    printf("Enter command : ");
    scanf("%s",x);
    if(st->top==st->size-1)
    printf("Stack full\n");
    else
    {
       st->top++;
       strcpy(st->c[st->top],x);
    }
}
void pop(struct Stack *st)
{
    char x[10];
    if(st->top==-1)
    printf("Stack is empty\n");
    else
    {
        strcpy(x,st->c[st->top]);
        st->top--;
    }
    printf("Command :%s has been removed\n",x);
}
void display(struct Stack st)
{
    for(int i=st.top;i>=0;i--)
```

```
    {
        printf("%s\n",st.c[i]);
    }
}
void  top(struct Stack st)
{
    if(st.top==-1)
    printf("Stack is empty\n");
    else
    printf("Top command is %s\n",st.c[st.top]);
}
int peek(struct Stack st)
{
    char x[10];
    printf("Enter the command to search :");
    scanf("%s",x);
    if(st.top==-1)
    printf("Stack is empty\n");
    else
    {
        for(int i=st.top;i>=0;i--)
        {
            if(strcmp(st.c[i],x)==0)
            return 1;
        }
    }
    return 0;
}
/*Implement a stack to handle events in an aerospace simulation. Include a
switch-case menu with options:
1: Push a new event
2: Pop the last event
3: Display all events
4: Peek at the most recent event
5: Search for a specific event
6: Exit*/
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
struct Stack
```

```
{
int size;
int top;
char **c;
};
void create(struct Stack *);
void push(struct Stack *);
void pop(struct Stack *);
void display(struct Stack);
void top(struct Stack);
int peek(struct Stack);
void main()
{
    struct Stack st;
    create(&st);
    int choice,f;
    do
    {
       printf("1: Push a new event\n");
       printf("2: Pop the last event\n");
       printf("3: Display all events\n");
       printf("4: Peek at the most recent event\n");
       printf("5: Search for a specific event\n");
       printf("6: Exit\n");
       printf("Enter choice : ");
       scanf("%d",&choice);
       switch(choice)
       {
        case 1:push(&st);
        break;
        case 2:pop(&st);
        break;
        case 3:display(st);
        break;
        case 4:top(st);
        break;
        case 5:f=peek(st);
        if(f==0)
        printf("No event found\n");
        else
```

```
        printf("Event found\n");
        break;
        case 6:printf("Exiting.....\n");
        break;
        default :printf("Enter valid option\n");
        break;
       }
    } while (choice!=6);

}
void create(struct Stack *st){
    printf("Enter The Size : ");
    scanf("%d",&st->size);
    st->top = -1;
    st->c = (char **)malloc((st->size) * sizeof(char*));
    for(int i=0;i<st->size;i++)
    st->c[i]=(char *)malloc(10*sizeof(char));
}
void push(struct Stack *st)
{
    char x[10];
    printf("Enter event name : ");
    scanf("%s",x);
    if(st->top==st->size-1)
    printf("Stack full\n");
    else
    {
       st->top++;
       strcpy(st->c[st->top],x);
    }
}
void pop(struct Stack *st)
{
    char x[10];
    if(st->top==-1)
    printf("Stack is empty\n");
    else
    {
        strcpy(x,st->c[st->top]);
        st->top--;
```

```
    }
    printf("Event :%s has been removed\n",x);
}
void display(struct Stack st)
{
    for(int i=st.top;i>=0;i--)
    {
        printf("%s\n",st.c[i]);
    }
}
void  top(struct Stack st)
{
    if(st.top==-1)
    printf("Stack is empty\n");
    else
    printf("Last event is %s\n",st.c[st.top]);
}
int peek(struct Stack st)
{
    char x[10];
    printf("Enter the event name to search :");
    scanf("%s",x);
    if(st.top==-1)
    printf("Stack is empty\n");
    else
    {
        for(int i=st.top;i>=0;i--)
        {
            if(strcmp(st.c[i],x)==0)
            return 1;
        }
    }
    return 0;
}
/*Use a stack to keep track of training maneuvers for pilots. Implement a
switch-case menu with options:
1: Add a maneuver (push)
2: Remove the last maneuver (pop)
3: View all maneuvers
4: Peek at the most recent maneuver
```

```c
5: Search for a specific maneuver
6: Exit*/
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
struct Stack
{
int size;
int top;
char **c;
};
void create(struct Stack *);
void push(struct Stack *);
void pop(struct Stack *);
void display(struct Stack);
void top(struct Stack);
int peek(struct Stack);
void main()
{
    struct Stack st;
    create(&st);
    int choice,f;
    do
    {
       printf("1: Add a maneuver (push)\n");
       printf("2: Remove the last maneuver (pop)\n");
       printf("3: View all maneuvers\n");
       printf("4: Peek at the most recent maneuver\n");
       printf("5: Search for a specific maneuver\n");
       printf("6: Exit\n");
       printf("Enter choice : ");
       scanf("%d",&choice);
       switch(choice)
       {
        case 1:push(&st);
        break;
        case 2:pop(&st);
        break;
        case 3:display(st);
        break;
```

```
        case 4:top(st);
        break;
        case 5:f=peek(st);
        if(f==0)
        printf("No maneuver found\n");
        else
        printf("Maneuver found\n");
        break;
        case 6:printf("Exiting.....\n");
        break;
        default :printf("Enter valid option\n");
        break;
       }
    } while (choice!=6);

}
void create(struct Stack *st){
    printf("Enter The Size : ");
    scanf("%d",&st->size);
    st->top = -1;
    st->c = (char **)malloc((st->size) * sizeof(char*));
    for(int i=0;i<st->size;i++)
    st->c[i]=(char *)malloc(10*sizeof(char));
}
void push(struct Stack *st)
{
    char x[10];
    printf("Enter maneuver : ");
    scanf("%s",x);
    if(st->top==st->size-1)
    printf("Stack full\n");
    else
    {
       st->top++;
       strcpy(st->c[st->top],x);
    }
}
void pop(struct Stack *st)
{
    char x[10];
```

```
    if(st->top==-1)
    printf("Stack is empty\n");
    else
    {
        strcpy(x,st->c[st->top]);
        st->top--;
    }
    printf("Maneuver :%s has been removed\n",x);
}
void display(struct Stack st)
{
    for(int i=st.top;i>=0;i--)
    {
        printf("%s\n",st.c[i]);
    }
}
void  top(struct Stack st)
{
    if(st.top==-1)
    printf("Stack is empty\n");
    else
    printf("Last maneuver is %s\n",st.c[st.top]);
}
int peek(struct Stack st)
{
    char x[10];
    printf("Enter the maneuver to search :");
    scanf("%s",x);
    if(st.top==-1)
    printf("Stack is empty\n");
    else
    {
        for(int i=st.top;i>=0;i--)
        {
            if(strcmp(st.c[i],x)==0)
            return 1;
        }
    }
    return 0;
}
```

```c
/*Use a stack to keep track of training maneuvers for pilots. Implement a
switch-case menu with options:
1: Add a maneuver (push)
2: Remove the last maneuver (pop)
3: View all maneuvers
4: Peek at the most recent maneuver
5: Search for a specific maneuver
6: Exit*/
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
struct Stack
{
int size;
int top;
char **c;
};
void create(struct Stack *);
void push(struct Stack *);
void pop(struct Stack *);
void display(struct Stack);
void top(struct Stack);
int peek(struct Stack);
void main()
{
    struct Stack st;
    create(&st);
    int choice,f;
    do
    {
       printf("1: Add a maneuver (push)\n");
       printf("2: Remove the last maneuver (pop)\n");
       printf("3: View all maneuvers\n");
       printf("4: Peek at the most recent maneuver\n");
       printf("5: Search for a specific maneuver\n");
       printf("6: Exit\n");
       printf("Enter choice : ");
       scanf("%d",&choice);
       switch(choice)
       {
```

```
        case 1:push(&st);
        break;
        case 2:pop(&st);
        break;
        case 3:display(st);
        break;
        case 4:top(st);
        break;
        case 5:f=peek(st);
        if(f==0)
        printf("No maneuver found\n");
        else
        printf("Maneuver found\n");
        break;
        case 6:printf("Exiting.....\n");
        break;
        default :printf("Enter valid option\n");
        break;
       }
    } while (choice!=6);

}
void create(struct Stack *st){
    printf("Enter The Size : ");
    scanf("%d",&st->size);
    st->top = -1;
    st->c = (char **)malloc((st->size) * sizeof(char*));
    for(int i=0;i<st->size;i++)
    st->c[i]=(char *)malloc(10*sizeof(char));
}
void push(struct Stack *st)
{
    char x[10];
    printf("Enter maneuver : ");
    scanf("%s",x);
    if(st->top==st->size-1)
    printf("Stack full\n");
    else
    {
       st->top++;
```

```
        strcpy(st->c[st->top],x);
    }
}
void pop(struct Stack *st)
{
    char x[10];
    if(st->top==-1)
    printf("Stack is empty\n");
    else
    {
        strcpy(x,st->c[st->top]);
        st->top--;
    }
    printf("Maneuver :%s has been removed\n",x);
}
void display(struct Stack st)
{
    for(int i=st.top;i>=0;i--)
    {
        printf("%s\n",st.c[i]);
    }
}
void  top(struct Stack st)
{
    if(st.top==-1)
    printf("Stack is empty\n");
    else
    printf("Last maneuver is %s\n",st.c[st.top]);
}
int peek(struct Stack st)
{
    char x[10];
    printf("Enter the maneuver to search :");
    scanf("%s",x);
    if(st.top==-1)
    printf("Stack is empty\n");
    else
    {
        for(int i=st.top;i>=0;i--)
        {
```

```
            if(strcmp(st.c[i],x)==0)
            return 1;
        }
    }
    return 0;
}
/*Create a stack-based system for handling emergency procedures in a
spacecraft. Implement a switch-case menu with options:
1: Add a procedure (push)
2: Remove the last procedure (pop)
3: View all procedures
4: Peek at the next procedure
5: Search for a specific procedure
6: Exit*/
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
struct Stack
{
int size;
int top;
char **c;
};
void create(struct Stack *);
void push(struct Stack *);
void pop(struct Stack *);
void display(struct Stack);
void top(struct Stack);
int peek(struct Stack);
void main()
{
    struct Stack st;
    create(&st);
    int choice,f;
    do
    {
       printf("1: Add a procedure (push)\n");
       printf("2: Remove the last procedure (pop)\n");
       printf("3: View all procedures\n");
       printf("4: Peek at the next procedure\n");
```

```
       printf("5: Search for a specific procedure\n");
       printf("6: Exit\n");
       printf("Enter choice : ");
       scanf("%d",&choice);
       switch(choice)
       {
        case 1:push(&st);
        break;
        case 2:pop(&st);
        break;
        case 3:display(st);
        break;
        case 4:top(st);
        break;
        case 5:f=peek(st);
        if(f==0)
        printf("No procedure found\n");
        else
        printf("Procedure found\n");
        break;
        case 6:printf("Exiting.....\n");
        break;
        default :printf("Enter valid option\n");
        break;
       }
    } while (choice!=6);

}
void create(struct Stack *st){
    printf("Enter The Size : ");
    scanf("%d",&st->size);
    st->top = -1;
    st->c = (char **)malloc((st->size) * sizeof(char*));
    for(int i=0;i<st->size;i++)
    st->c[i]=(char *)malloc(10*sizeof(char));
}
void push(struct Stack *st)
{
    char x[10];
    printf("Enter procedure : ");
```

```
    scanf("%s",x);
    if(st->top==st->size-1)
    printf("Stack full\n");
    else
    {
       st->top++;
       strcpy(st->c[st->top],x);
    }
}
void pop(struct Stack *st)
{
    char x[10];
    if(st->top==-1)
    printf("Stack is empty\n");
    else
    {
        strcpy(x,st->c[st->top]);
        st->top--;
    }
    printf("Procedure :%s has been removed\n",x);
}
void display(struct Stack st)
{
    for(int i=st.top;i>=0;i--)
    {
        printf("%s\n",st.c[i]);
    }
}
void  top(struct Stack st)
{
    if(st.top==-1)
    printf("Stack is empty\n");
    else
    printf("Last procedure is %s\n",st.c[st.top]);
}
int peek(struct Stack st)
{
    char x[10];
    printf("Enter the procedure to search :");
    scanf("%s",x);
```

```
    if(st.top==-1)
    printf("Stack is empty\n");
    else
    {
        for(int i=st.top;i>=0;i--)
        {
            if(strcmp(st.c[i],x)==0)
            return 1;
        }
    }
    return 0;
}
/*Implement a stack for logging astronaut activities during a mission. Use
a switch-case menu with options:
1: Add a new activity (push)
2: Remove the last activity (pop)
3: Display the activity log
4: Peek at the most recent activity
5: Search for a specific activity
6: Exit*/
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
struct Stack
{
int size;
int top;
char **c;
};
void create(struct Stack *);
void push(struct Stack *);
void pop(struct Stack *);
void display(struct Stack);
void top(struct Stack);
int peek(struct Stack);
void main()
{
    struct Stack st;
    create(&st);
    int choice,f;
```

```
    do
    {
       printf("1: Add a new activity (push)\n");
       printf("2: Remove the last activity (pop)\n");
       printf("3: Display the activity log\n");
       printf("4: Peek at the most recent activity\n");
       printf("5: Search for a specific activity\n");
       printf("6: Exit\n");
       printf("Enter choice : ");
       scanf("%d",&choice);
       switch(choice)
       {
        case 1:push(&st);
        break;
        case 2:pop(&st);
        break;
        case 3:display(st);
        break;
        case 4:top(st);
        break;
        case 5:f=peek(st);
        if(f==0)
        printf("No activity found\n");
        else
        printf("Activity found\n");
        break;
        case 6:printf("Exiting.....\n");
        break;
        default :printf("Enter valid option\n");
        break;
       }
    } while (choice!=6);

}
void create(struct Stack *st){
    printf("Enter The Size : ");
    scanf("%d",&st->size);
    st->top = -1;
    st->c = (char **)malloc((st->size) * sizeof(char*));
    for(int i=0;i<st->size;i++)
```

```
    st->c[i]=(char *)malloc(10*sizeof(char));
}
void push(struct Stack *st)
{
    char x[10];
    printf("Enter activity : ");
    scanf("%s",x);
    if(st->top==st->size-1)
    printf("Stack full\n");
    else
    {
       st->top++;
       strcpy(st->c[st->top],x);
    }
}
void pop(struct Stack *st)
{
    char x[10];
    if(st->top==-1)
    printf("Stack is empty\n");
    else
    {
        strcpy(x,st->c[st->top]);
        st->top--;
    }
    printf("Activity :%s has been removed\n",x);
}
void display(struct Stack st)
{
    for(int i=st.top;i>=0;i--)
    {
        printf("%s\n",st.c[i]);
    }
}
void  top(struct Stack st)
{
    if(st.top==-1)
    printf("Stack is empty\n");
    else
    printf("Last activity is %s\n",st.c[st.top]);
```

```c
}
int peek(struct Stack st)
{
    char x[10];
    printf("Enter the activity to search :");
    scanf("%s",x);
    if(st.top==-1)
    printf("Stack is empty\n");
    else
    {
        for(int i=st.top;i>=0;i--)
        {
            if(strcmp(st.c[i],x)==0)
            return 1;
        }
    }
    return 0;
}
/*Develop a stack to monitor fuel usage in an aerospace vehicle. Implement
a switch-case menu with options:
1: Add a fuel usage entry (push)
2: Remove the last entry (pop)
3: View all fuel usage data
4: Peek at the latest fuel usage entry
5: Search for a specific fuel usage entry
6: Exit*/
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
struct Stack
{
int size;
int top;
float *f;
};
void create(struct Stack *);
void push(struct Stack *);
void pop(struct Stack *);
void display(struct Stack);
void top(struct Stack);
```

```c
int peek(struct Stack);
void main()
{
    struct Stack st;
    create(&st);
    int choice,f;
    do
    {
       printf("1: Add a fuel usage entry (push)\n");
       printf("2: Remove the last entry (pop)\n");
       printf("3: View all fuel usage data\n");
       printf("4: Peek at the latest fuel usage entry\n");
       printf("5: Search for a specific fuel usage entry\n");
       printf("6: Exit\n");
       printf("Enter choice : ");
       scanf("%d",&choice);
       switch(choice)
       {
        case 1:push(&st);
        break;
        case 2:pop(&st);
        break;
        case 3:display(st);
        break;
        case 4:top(st);
        break;
        case 5:f=peek(st);
        if(f==0)
        printf("No fuel reading found\n");
        else
        printf("Fuel reading found\n");
        break;
        case 6:printf("Exiting.....\n");
        break;
        default :printf("Enter valid option\n");
        break;
       }
    } while (choice!=6);

}
```

```
void create(struct Stack *st){
    printf("Enter The Size : ");
    scanf("%d",&st->size);
    st->top = -1;
    st->f = (float *)malloc((st->size) * sizeof(float));
}
void push(struct Stack *st)
{
    float x;
    printf("Enter fuel reading : ");
    scanf("%f",&x);
    if(st->top==st->size-1)
    printf("Stack full\n");
    else
    {
       st->top++;
       st->f[st->top]=x;
    }
}
void pop(struct Stack *st)
{
   float x;
    if(st->top==-1)
    printf("Stack is empty\n");
    else
    {
        x=st->f[st->top];
        st->top--;
    }
    printf("fuel reading :%s has been removed\n",x);
}
void display(struct Stack st)
{
    for(int i=st.top;i>=0;i--)
    {
        printf("%.2f\n",st.f[i]);
    }
}
void  top(struct Stack st)
{
```

```c
    if(st.top==-1)
    printf("Stack is empty\n");
    else
    printf("Last fuel reading is %.2f\n",st.f[st.top]);
}
int peek(struct Stack st)
{
    float x;
    printf("Enter the fuel reading to search :");
    scanf("%f",&x);
    if(st.top==-1)
    printf("Stack is empty\n");
    else
    {
        for(int i=st.top;i>=0;i--)
        {
            if(st.f[i]==x)
            return 1;
        }
    }
    return 0;
}
/*: Implement a stack-based system using a linked list to manage order
processing. Use a switch-case menu with options:
1: Add a new order (push)
2: Process the last order (pop)
3: Display all pending orders
4: Peek at the next order to be processed
5: Search for a specific order
6: Exit*/
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
struct Stack
{
    int id;
    struct Stack *next;
}*head=NULL;
void push();
void pop();
```

```c
void display();
void top();
int peek();
void main()
{
    int choice,f;
    do
    {
       printf("1: Add a new order (push)\n");
       printf("2: Process the last order (pop)\n");
       printf("3: Display all pending orders\n");
       printf("4: Peek at the next order to be processed\n");
       printf("5: Search for a specific order\n");
       printf("6: Exit\n");
       printf("Enter choice : ");
       scanf("%d",&choice);
       switch(choice)
       {
        case 1:push();
        break;
        case 2:pop();
        break;
        case 3:display();
        break;
        case 4:top();
        break;
        case 5:f=peek();
        if(f==0)
        printf("No order found\n");
        else
        printf("Order found\n");
        break;
        case 6:printf("Exiting.....\n");
        break;
        default :printf("Enter valid option\n");
        break;
       }
    } while (choice!=6);

}
```

```
void push()
{
    int n;
    struct Stack *t=(struct Stack *)malloc(sizeof(struct Stack));
    if(t==NULL)
    printf("Stack full\n");
    else
    {
         printf("Enter Order id : ");
         scanf("%d",&n);
         t->next=head;
         t->id=n;
         head=t;
    }
}
void pop()
{
    int n;
    struct Stack *t;
    if(head==NULL)
    printf("Stack is empty\n");
    else
    {
        t=head;
        head=head->next;
        n=t->id;
        printf("Order %d removed from stack\n",n);
        free(t);
    }

}
void display()
{
    struct Stack*t;
    t=head;
    while(t!=NULL)
    {
        printf("%d\n",t->id);
        t=t->next;
    }
```

```c
}
void top()
{
    if(head==NULL)
    printf("Stack is empty\n");
    else
    {
        printf("Recent order : %d\n",head->id);
    }
}
int peek()
{
    int n;
    struct Stack *t;
    t=head;
    printf("Enter order number : ");
    scanf("%d",&n);
    if(head==NULL)
    printf("Stack is empty\n");
    else
    {
        while(t!=NULL)
        {
            if(t->id==n)
            return 1;
            t=t->next;
        }
    }
    return 0;
}
/*Create a stack using a linked list to handle customer support tickets.
Include a switch-case menu with options:
1: Add a new ticket (push)
2: Resolve the latest ticket (pop)
3: View all pending tickets
4: Peek at the latest ticket
5: Search for a specific ticket
6: Exit*/
#include<stdio.h>
#include<stdlib.h>
```

```c
#include<string.h>
struct Stack
{
    int id;
    struct Stack *next;
}*head=NULL;
void push();
void pop();
void display();
void top();
int peek();
void main()
{
    int choice,f;
    do
    {
       printf("1: Add a new ticket (push)\n");
       printf("2: Resolve the latest ticket (pop)\n");
       printf("3: View all pending tickets\n");
       printf("4: Peek at the latest ticket\n");
       printf("5: Search for a specific ticket\n");
       printf("6: Exit\n");
       printf("Enter choice : ");
       scanf("%d",&choice);
       switch(choice)
       {
        case 1:push();
        break;
        case 2:pop();
        break;
        case 3:display();
        break;
        case 4:top();
        break;
        case 5:f=peek();
        if(f==0)
        printf("No ticket found\n");
        else
        printf("Ticket found\n");
        break;
```

```
        case 6:printf("Exiting.....\n");
        break;
        default :printf("Enter valid option\n");
        break;
       }
    } while (choice!=6);

}
void push()
{
    int n;
    struct Stack *t=(struct Stack *)malloc(sizeof(struct Stack));
    if(t==NULL)
    printf("Stack full\n");
    else
    {
         printf("Enter ticket no : ");
         scanf("%d",&n);
         t->next=head;
         t->id=n;
         head=t;
    }
}
void pop()
{
    int n;
    struct Stack *t;
    if(head==NULL)
    printf("Stack is empty\n");
    else
    {
        t=head;
        head=head->next;
        n=t->id;
        printf("ticket number %d removed from stack\n",n);
        free(t);
    }

}
void display()
```

```c
{
    struct Stack*t;
    t=head;
    while(t!=NULL)
    {
        printf("%d\n",t->id);
        t=t->next;
    }
}
void top()
{
    if(head==NULL)
    printf("Stack is empty\n");
    else
    {
        printf("Recent ticket number : %d\n",head->id);
    }
}
int peek()
{
    int n;
    struct Stack *t;
    t=head;
    printf("Enter ticket number : ");
    scanf("%d",&n);
    if(head==NULL)
    printf("Stack is empty\n");
    else
    {
        while(t!=NULL)
        {
            if(t->id==n)
            return 1;
            t=t->next;
        }
    }
    return 0;
}
/*Develop a stack to manage product returns using a linked list. Implement
a switch-case menu with options:
```

```c
/*1: Add a new return request (push)
2: Process the last return (pop)
3: Display all return requests
4: Peek at the next return to process
5: Search for a specific return request
6: Exit*/
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
struct Stack
{
    int id;
    struct Stack *next;
}*head=NULL;
void push();
void pop();
void display();
void top();
int peek();
void main()
{
    int choice,f;
    do
    {
       printf("1: Add a new return request (push)\n");
       printf("2: Process the last return (pop)\n");
       printf("3: Display all return requests\n");
       printf("4: Peek at the next return to process\n");
       printf("5: Search for a specific return request\n");
       printf("6: Exit\n");
       printf("Enter choice : ");
       scanf("%d",&choice);
       switch(choice)
       {
        case 1:push();
        break;
        case 2:pop();
        break;
        case 3:display();
        break;
```

```
        case 4:top();
        break;
        case 5:f=peek();
        if(f==0)
        printf("No request found\n");
        else
        printf("Request found\n");
        break;
        case 6:printf("Exiting.....\n");
        break;
        default :printf("Enter valid option\n");
        break;
       }
    } while (choice!=6);

}
void push()
{
    int n;
    struct Stack *t=(struct Stack *)malloc(sizeof(struct Stack));
    if(t==NULL)
    printf("Stack full\n");
    else
    {
         printf("Enter request id : ");
         scanf("%d",&n);
         t->next=head;
         t->id=n;
         head=t;
    }
}
void pop()
{
    int n;
    struct Stack *t;
    if(head==NULL)
    printf("Stack is empty\n");
    else
    {
        t=head;
```

```
        head=head->next;
        n=t->id;
        printf("Request id %d removed from stack\n",n);
        free(t);
    }

}
void display()
{
    struct Stack*t;
    t=head;
    while(t!=NULL)
    {
        printf("%d\n",t->id);
        t=t->next;
    }
}
void top()
{
    if(head==NULL)
    printf("Stack is empty\n");
    else
    {
        printf("Request id : %d\n",head->id);
    }
}
int peek()
{
    int n;
    struct Stack *t;
    t=head;
    printf("Enter request id : ");
    scanf("%d",&n);
    if(head==NULL)
    printf("Stack is empty\n");
    else
    {
        while(t!=NULL)
        {
            if(t->id==n)
```

```
            return 1;
            t=t->next;
        }
    }
    return 0;
}
/*Implement a stack to manage inventory restocking using a linked list.
Use a switch-case menu with options:
1: Add a restock entry (push)
2: Process the last restock (pop)
3: View all restock entries
4: Peek at the latest restock entry
5: Search for a specific restock entry
6: Exit*/
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
struct Stack
{
    char name[10];
    struct Stack *next;
}*head=NULL;
void push();
void pop();
void display();
void top();
int peek();
void main()
{
    int choice,f;
    do
    {
       printf("1: Add a restock entry (push)\n");
       printf("2: Process the last restock (pop)\n");
       printf("3: View all restock entries\n");
       printf("4: Peek at the latest restock entry\n");
       printf("5: Search for a specific restock entry\n");
       printf("6: Exit\n");
       printf("Enter choice : ");
       scanf("%d",&choice);
```

```
       switch(choice)
       {
        case 1:push();
        break;
        case 2:pop();
        break;
        case 3:display();
        break;
        case 4:top();
        break;
        case 5:f=peek();
        if(f==0)
        printf("No request found\n");
        else
        printf("Request found\n");
        break;
        case 6:printf("Exiting.....\n");
        break;
        default :printf("Enter valid option\n");
        break;
       }
    } while (choice!=6);

}
void push()
{
    char n[10];
    struct Stack *t=(struct Stack *)malloc(sizeof(struct Stack));
    if(t==NULL)
    printf("Stack full\n");
    else
    {
         printf("Enter request item : ");
         scanf("%s",n);
         t->next=head;
         strcpy(t->name,n);
         head=t;
    }
}
void pop()
```

```
{
    char n[10];
    struct Stack *t;
    if(head==NULL)
    printf("Stack is empty\n");
    else
    {
        t=head;
        head=head->next;
        strcpy(n,t->name);
        printf("Request id %s removed from stack\n",n);
        free(t);
    }

}
void display()
{
    struct Stack*t;
    t=head;
    while(t!=NULL)
    {
        printf("%s\n",t->name);
        t=t->next;
    }
}
void top()
{
    if(head==NULL)
    printf("Stack is empty\n");
    else
    {
        printf("Request item : %s\n",head->name);
    }
}
int peek()
{
    char n[10];
    struct Stack *t;
    t=head;
    printf("Enter request name : ");
```

```
    scanf("%s",n);
    if(head==NULL)
    printf("Stack is empty\n");
    else
    {
        while(t!=NULL)
        {
            if(strcmp(t->name,n)==0)
            return 1;
            t=t->next;
        }
    }
    return 0;
}
/*
Create a stack for managing flash sale deals using a linked list. Include
a switch-case menu with options:
1: Add a new deal (push)
2: Remove the last deal (pop)
3: View all active deals
4: Peek at the latest deal
5: Search for a specific deal
6: Exit
*/
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
struct Stack
{
    int percent;
    struct Stack *next;
}*head=NULL;
void push();
void pop();
void display();
void top();
int peek();
void main()
{
    int choice,f;
```

```c
    do
    {
       printf("1: Add a new deal (push)\n");
       printf("2: Remove the last deal (pop)\n");
       printf("3: View all active deals\n");
       printf("4: Peek at the latest deal\n");
       printf("5: Search for a specific deal\n");
       printf("6: Exit\n");
       printf("Enter choice : ");
       scanf("%d",&choice);
       switch(choice)
       {
        case 1:push();
        break;
        case 2:pop();
        break;
        case 3:display();
        break;
        case 4:top();
        break;
        case 5:f=peek();
        if(f==0)
        printf("No deal found\n");
        else
        printf("Deal found\n");
        break;
        case 6:printf("Exiting.....\n");
        break;
        default :printf("Enter valid option\n");
        break;
       }
    } while (choice!=6);

}
void push()
{
    int n;
    struct Stack *t=(struct Stack *)malloc(sizeof(struct Stack));
    if(t==NULL)
    printf("Stack full\n");
```

```
    else
    {
         printf("Enter deal : ");
         scanf("%d",&n);
         t->next=head;
         t->percent=n;
         head=t;
    }
}
void pop()
{
    int n;
    struct Stack *t;
    if(head==NULL)
    printf("Stack is empty\n");
    else
    {
        t=head;
        head=head->next;
        n=t->percent;
        printf("Deal %d removed from stack\n",n);
        free(t);
    }

}
void display()
{
    struct Stack*t;
    t=head;
    while(t!=NULL)
    {
        printf("%d\n",t->percent);
        t=t->next;
    }
}
void top()
{
    if(head==NULL)
    printf("Stack is empty\n");
    else
```

```
    {
        printf("Deal : %d\n",head->percent);
    }
}
int peek()
{
    int n;
    struct Stack *t;
    t=head;
    printf("Enter deal : ");
    scanf("%d",&n);
    if(head==NULL)
    printf("Stack is empty\n");
    else
    {
        while(t!=NULL)
        {
            if(t->percent==n)
            return 1;
            t=t->next;
        }
    }
    return 0;
}
/*Use a stack to track user session history in an e-commerce site using a
linked list. Implement a switch-case menu with options:
1: Add a session (push)
2: End the last session (pop)
3: Display all sessions
4: Peek at the most recent session
5: Search for a specific session
6: Exit*/
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
struct Stack
{
    char name[10];
    struct Stack *next;
}*head=NULL;
```

```c
void push();
void pop();
void display();
void top();
int peek();
void main()
{
    int choice,f;
    do
    {
       printf("1:Add a session (push)\n");
       printf("2: End the last session (pop)\n");
       printf("3: Display all sessions\n");
       printf("4: Peek at the most recent session\n");
       printf("5: Search for a specific session\n");
       printf("6: Exit\n");
       printf("Enter choice : ");
       scanf("%d",&choice);
       switch(choice)
       {
        case 1:push();
        break;
        case 2:pop();
        break;
        case 3:display();
        break;
        case 4:top();
        break;
        case 5:f=peek();
        if(f==0)
        printf("No id found\n");
        else
        printf("Id found\n");
        break;
        case 6:printf("Exiting.....\n");
        break;
        default :printf("Enter valid option\n");
        break;
       }
    } while (choice!=6);
```

```
}
void push()
{
    char n[10];
    struct Stack *t=(struct Stack *)malloc(sizeof(struct Stack));
    if(t==NULL)
    printf("Stack full\n");
    else
    {
         printf("Enter user id : ");
         scanf("%s",n);
         t->next=head;
         strcpy(t->name,n);
         head=t;
    }
}
void pop()
{
    char n[10];
    struct Stack *t;
    if(head==NULL)
    printf("Stack is empty\n");
    else
    {
        t=head;
        head=head->next;
        strcpy(n,t->name);
        printf("User id %s removed from stack\n",n);
        free(t);
    }

}
void display()
{
    struct Stack*t;
    t=head;
    while(t!=NULL)
    {
        printf("%s\n",t->name);
```

```
        t=t->next;
    }
}
void top()
{
    if(head==NULL)
    printf("Stack is empty\n");
    else
    {
        printf("Recent user id : %s\n",head->name);
    }
}
int peek()
{
    char n[10];
    struct Stack *t;
    t=head;
    printf("Enter user id: ");
    scanf("%s",n);
    if(head==NULL)
    printf("Stack is empty\n");
    else
    {
        while(t!=NULL)
        {
            if(strcmp(t->name,n)==0)
            return 1;
            t=t->next;
        }
    }
    return 0;
}
/*Develop a stack to manage user wishlists using a linked list. Use a
switch-case menu with options:
1: Add a product to wishlist (push)
2: Remove the last added product (pop)
3: View all wishlist items
4: Peek at the most recent wishlist item
5: Search for a specific product in wishlist
6: Exi*/
```

```c
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
struct Stack
{
    char name[10];
    struct Stack *next;
}*head=NULL;
void push();
void pop();
void display();
void top();
int peek();
void main()
{
    int choice,f;
    do
    {
       printf("1: Add a product to wishlist (push)\n");
       printf("2: Remove the last added product (pop)\n");
       printf("3: View all wishlist items\n");
       printf("4: Peek at the most recent wishlist item\n");
       printf("5: Search for a specific product in wishlist\n");
       printf("6: Exit\n");
       printf("Enter choice : ");
       scanf("%d",&choice);
       switch(choice)
       {
        case 1:push();
        break;
        case 2:pop();
        break;
        case 3:display();
        break;
        case 4:top();
        break;
        case 5:f=peek();
        if(f==0)
        printf("No request found\n");
        else
```

```
        printf("Request found\n");
        break;
        case 6:printf("Exiting.....\n");
        break;
        default :printf("Enter valid option\n");
        break;
       }
    } while (choice!=6);

}
void push()
{
    char n[10];
    struct Stack *t=(struct Stack *)malloc(sizeof(struct Stack));
    if(t==NULL)
    printf("Stack full\n");
    else
    {
         printf("Enter request item : ");
         scanf("%s",n);
         t->next=head;
         strcpy(t->name,n);
         head=t;
    }
}
void pop()
{
    char n[10];
    struct Stack *t;
    if(head==NULL)
    printf("Stack is empty\n");
    else
    {
        t=head;
        head=head->next;
        strcpy(n,t->name);
        printf("Request id %s removed from stack\n",n);
        free(t);
    }
```

```
}
void display()
{
    struct Stack*t;
    t=head;
    while(t!=NULL)
    {
        printf("%s\n",t->name);
        t=t->next;
    }
}
void top()
{
    if(head==NULL)
    printf("Stack is empty\n");
    else
    {
        printf("Request item : %s\n",head->name);
    }
}
int peek()
{
    char n[10];
    struct Stack *t;
    t=head;
    printf("Enter request name : ");
    scanf("%s",n);
    if(head==NULL)
    printf("Stack is empty\n");
    else
    {
        while(t!=NULL)
        {
            if(strcmp(t->name,n)==0)
            return 1;
            t=t->next;
        }
    }
    return 0;
}
```

```c
/*mplement a stack to manage steps in the checkout process using a linked
list. Include a switch-case menu with options:
1: Add a checkout step (push)
2: Remove the last step (pop)
3: Display all checkout steps
4: Peek at the current step
5: Search for a specific step
6: Exit*/
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
struct Stack
{
    int step;
    struct Stack *next;
}*head=NULL;
void push();
void pop();
void display();
void top();
int peek();
void main()
{
    int choice,f;
    do
    {
       printf("1: Add a checkout step (push)\n");
       printf("2: Remove the last step (pop)\n");
       printf("3: Display all checkout steps\n");
       printf("4: Peek at the current step\n");
       printf("5: Search for a specific step\n");
       printf("6: Exit\n");
       printf("Enter choice : ");
       scanf("%d",&choice);
       switch(choice)
       {
        case 1:push();
        break;
        case 2:pop();
        break;
```

```
        case 3:display();
        break;
        case 4:top();
        break;
        case 5:f=peek();
        if(f==0)
        printf("No step found\n");
        else
        printf("Step found\n");
        break;
        case 6:printf("Exiting.....\n");
        break;
        default :printf("Enter valid option\n");
        break;
       }
    } while (choice!=6);

}
void push()
{
    int n;
    struct Stack *t=(struct Stack *)malloc(sizeof(struct Stack));
    if(t==NULL)
    printf("Stack full\n");
    else
    {
         printf("Enter deal : ");
         scanf("%d",&n);
         t->next=head;
         t->step=n;
         head=t;
    }
}
void pop()
{
    int n;
    struct Stack *t;
    if(head==NULL)
    printf("Stack is empty\n");
    else
```

```
    {
        t=head;
        head=head->next;
        n=t->step;
        printf("Step %d removed from stack\n",n);
        free(t);
    }

}
void display()
{
    struct Stack*t;
    t=head;
    while(t!=NULL)
    {
        printf("%d\n",t->step);
        t=t->next;
    }
}
void top()
{
    if(head==NULL)
    printf("Stack is empty\n");
    else
    {
        printf("Step : %d\n",head->step);
    }
}
int peek()
{
    int n;
    struct Stack *t;
    t=head;
    printf("Enter Step : ");
    scanf("%d",&n);
    if(head==NULL)
    printf("Stack is empty\n");
    else
    {
        while(t!=NULL)
```

```
        {
            if(t->step==n)
            return 1;
            t=t->next;
        }
    }
    return 0;
}
/*Create a stack for managing coupon codes using a linked list. Use a
switch-case menu with options:
1: Add a new coupon code (push)
2: Remove the last coupon code (pop)
3: View all available coupon codes
4: Peek at the latest coupon code
5: Search for a specific coupon code
6: Exit*/
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
struct Stack
{
    char code[10];
    struct Stack *next;
}*head=NULL;
void push();
void pop();
void display();
void top();
int peek();
void main()
{
    int choice,f;
    do
    {
       printf("1: Add a new coupon code (push)\n");
       printf("2: Remove the last coupon code (pop)\n");
       printf("3: View all available coupon codes\n");
       printf("4: Peek at the latest coupon code\n");
       printf("5: Search for a specific coupon code\n");
       printf("6: Exit\n");
```

```
       printf("Enter choice : ");
       scanf("%d",&choice);
       switch(choice)
       {
        case 1:push();
        break;
        case 2:pop();
        break;
        case 3:display();
        break;
        case 4:top();
        break;
        case 5:f=peek();
        if(f==0)
        printf("No code found\n");
        else
        printf("Coupon codde found\n");
        break;
        case 6:printf("Exiting.....\n");
        break;
        default :printf("Enter valid option\n");
        break;
       }
    } while (choice!=6);

}
void push()
{
    char n[10];
    struct Stack *t=(struct Stack *)malloc(sizeof(struct Stack));
    if(t==NULL)
    printf("Stack full\n");
    else
    {
         printf("Enter coupon code : ");
         scanf("%s",n);
         t->next=head;
         strcpy(t->code,n);
         head=t;
    }
```

```
}
void pop()
{
    char n[10];
    struct Stack *t;
    if(head==NULL)
    printf("Stack is empty\n");
    else
    {
        t=head;
        head=head->next;
        strcpy(n,t->code);
        printf("Coupon %s removed from stack\n",n);
        free(t);
    }

}
void display()
{
    struct Stack*t;
    t=head;
    while(t!=NULL)
    {
        printf("%s\n",t->code);
        t=t->next;
    }
}
void top()
{
    if(head==NULL)
    printf("Stack is empty\n");
    else
    {
        printf("Coupon : %s\n",head->code);
    }
}
int peek()
{
    char n[10];
    struct Stack *t;
```

```
    t=head;
    printf("Enter coupon code : ");
    scanf("%s",n);
    if(head==NULL)
    printf("Stack is empty\n");
    else
    {
        while(t!=NULL)
        {
            if(strcmp(t->code,n)==0)
            return 1;
            t=t->next;
        }
    }
    return 0;
}
/*Develop a stack to track shipping status updates using a linked list.
Implement a switch-case menu with options:
1: Add a shipping status update (push)
2: Remove the last update (pop)
3: View all shipping status updates
4: Peek at the latest update
5: Search for a specific update
6: Exit*/
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
struct Stack
{
    char status[10];
    struct Stack *next;
}*head=NULL;
void push();
void pop();
void display();
void top();
int peek();
void main()
{
    int choice,f;
```

```
    do
    {
       printf("1: Add a shipping status update (push)\n");
       printf("2: Remove the last update (pop)\n");
       printf("3: View all shipping status updates\n");
       printf("4: Peek at the latest update\n");
       printf("5: Search for a specific update\n");
       printf("6: Exit\n");
       printf("Enter choice : ");
       scanf("%d",&choice);
       switch(choice)
       {
        case 1:push();
        break;
        case 2:pop();
        break;
        case 3:display();
        break;
        case 4:top();
        break;
        case 5:f=peek();
        if(f==0)
        printf("No status found\n");
        else
        printf("Shipping status found\n");
        break;
        case 6:printf("Exiting.....\n");
        break;
        default :printf("Enter valid option\n");
        break;
       }
    } while (choice!=6);

}
void push()
{
    char n[10];
    struct Stack *t=(struct Stack *)malloc(sizeof(struct Stack));
    if(t==NULL)
    printf("Stack full\n");
```

```
    else
    {
         printf("Enter shipping status : ");
         scanf("%s",n);
         t->next=head;
         strcpy(t->status,n);
         head=t;
    }
}
void pop()
{
    char n[10];
    struct Stack *t;
    if(head==NULL)
    printf("Stack is empty\n");
    else
    {
        t=head;
        head=head->next;
        strcpy(n,t->status);
        printf("Shipping status %s removed from stack\n",n);
        free(t);
    }

}
void display()
{
    struct Stack*t;
    t=head;
    while(t!=NULL)
    {
        printf("%s\n",t->status);
        t=t->next;
    }
}
void top()
{
    if(head==NULL)
    printf("Stack is empty\n");
    else
```

```
    {
        printf("Shipping status : %s\n",head->status);
    }
}
int peek()
{
    char n[10];
    struct Stack *t;
    t=head;
    printf("Enter shipping status : ");
    scanf("%s",n);
    if(head==NULL)
    printf("Stack is empty\n");
    else
    {
        while(t!=NULL)
        {
            if(strcmp(t->status,n)==0)
            return 1;
            t=t->next;
        }
    }
    return 0;
}
/*Use a stack to manage user reviews for products using a linked list.
Include a switch-case menu with options:
1: Add a new review (push)
2: Remove the last review (pop)
3: Display all reviews
4: Peek at the latest review
5: Search for a specific review
6: Exit*/
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
struct Stack
{
    char review[10];
    struct Stack *next;
}*head=NULL;
```

```c
void push();
void pop();
void display();
void top();
int peek();
void main()
{
    int choice,f;
    do
    {
       printf("1: Add a new review (push)\n");
       printf("2: Remove the last review (pop)\n");
       printf("3: Display all reviews\n");
       printf("4: Peek at the latest review\n");
       printf("5: Search for a specific review\n");
       printf("6: Exit\n");
       printf("Enter choice : ");
       scanf("%d",&choice);
       switch(choice)
       {
        case 1:push();
        break;
        case 2:pop();
        break;
        case 3:display();
        break;
        case 4:top();
        break;
        case 5:f=peek();
        if(f==0)
        printf("No review found\n");
        else
        printf("Review found\n");
        break;
        case 6:printf("Exiting.....\n");
        break;
        default :printf("Enter valid option\n");
        break;
       }
    } while (choice!=6);
```

```
}
void push()
{
    char n[10];
    struct Stack *t=(struct Stack *)malloc(sizeof(struct Stack));
    if(t==NULL)
    printf("Stack full\n");
    else
    {
         printf("Enter review : ");
         scanf("%s",n);
         t->next=head;
         strcpy(t->review,n);
         head=t;
    }
}
void pop()
{
    char n[10];
    struct Stack *t;
    if(head==NULL)
    printf("Stack is empty\n");
    else
    {
        t=head;
        head=head->next;
        strcpy(n,t->review);
        printf("Review %s removed from stack\n",n);
        free(t);
    }

}
void display()
{
    struct Stack*t;
    t=head;
    while(t!=NULL)
    {
        printf("%s\n",t->review);
```

```
        t=t->next;
    }
}
void top()
{
    if(head==NULL)
    printf("Stack is empty\n");
    else
    {
        printf("Review : %s\n",head->review);
    }
}
int peek()
{
    char n[10];
    struct Stack *t;
    t=head;
    printf("Enter review : ");
    scanf("%s",n);
    if(head==NULL)
    printf("Stack is empty\n");
    else
    {
        while(t!=NULL)
        {
            if(strcmp(t->review,n)==0)
            return 1;
            t=t->next;
        }
    }
    return 0;
}
/*Create a stack for managing promotional notifications using a linked
list. Use a switch-case menu with options:
1: Add a new notification (push)
2: Remove the last notification (pop)
3: View all notifications
4: Peek at the latest notification
5: Search for a specific notification
6: Exit*/
```

```c
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
struct Stack
{
    char noti[10];
    struct Stack *next;
}*head=NULL;
void push();
void pop();
void display();
void top();
int peek();
void main()
{
    int choice,f;
    do
    {
       printf("1: Add a new notification (push))\n");
       printf("2: Remove the last notification (pop)\n");
       printf("3: View all notifications\n");
       printf("4: Peek at the latest notification\n");
       printf("5: Search for a specific notification\n");
       printf("6: Exit\n");
       printf("Enter choice : ");
       scanf("%d",&choice);
       switch(choice)
       {
        case 1:push();
        break;
        case 2:pop();
        break;
        case 3:display();
        break;
        case 4:top();
        break;
        case 5:f=peek();
        if(f==0)
        printf("No notification found\n");
        else
```

```
        printf("Notification found\n");
        break;
        case 6:printf("Exiting.....\n");
        break;
        default :printf("Enter valid option\n");
        break;
       }
    } while (choice!=6);

}
void push()
{
    char n[10];
    struct Stack *t=(struct Stack *)malloc(sizeof(struct Stack));
    if(t==NULL)
    printf("Stack full\n");
    else
    {
         printf("Enter notification : ");
         scanf("%s",n);
         t->next=head;
         strcpy(t->noti,n);
         head=t;
    }
}
void pop()
{
    char n[10];
    struct Stack *t;
    if(head==NULL)
    printf("Stack is empty\n");
    else
    {
        t=head;
        head=head->next;
        strcpy(n,t->noti);
        printf("Notification %s removed from stack\n",n);
        free(t);
    }
```

```
}
void display()
{
    struct Stack*t;
    t=head;
    while(t!=NULL)
    {
        printf("%s\n",t->noti);
        t=t->next;
    }
}
void top()
{
    if(head==NULL)
    printf("Stack is empty\n");
    else
    {
        printf("Notification : %s\n",head->noti);
    }
}
int peek()
{
    char n[10];
    struct Stack *t;
    t=head;
    printf("Enter notification : ");
    scanf("%s",n);
    if(head==NULL)
    printf("Stack is empty\n");
    else
    {
        while(t!=NULL)
        {
            if(strcmp(t->noti,n)==0)
            return 1;
            t=t->next;
        }
    }
    return 0;
}
```

```c
/*Implement a stack to track the viewing history of products using a
linked list. Include a switch-case menu with options:
1: Add a product to viewing history (push)
2: Remove the last viewed product (pop)
3: Display all viewed products
4: Peek at the most recent product viewed
5: Search for a specific product
6: Exit*/
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
struct Stack
{
    char name[10];
    struct Stack *next;
}*head=NULL;
void push();
void pop();
void display();
void top();
int peek();
void main()
{
    int choice,f;
    do
    {
       printf("1: Add a product to viewing history (push)\n");
       printf("2: Remove the last viewed product (pop)\n");
       printf("3: Display all viewed products\n");
       printf("4: Peek at the most recent wishlist item\n");
       printf("5: Search for a specific product\n");
       printf("6: Exit\n");
       printf("Enter choice : ");
       scanf("%d",&choice);
       switch(choice)
       {
        case 1:push();
        break;
        case 2:pop();
        break;
```

```
        case 3:display();
        break;
        case 4:top();
        break;
        case 5:f=peek();
        if(f==0)
        printf("No product found\n");
        else
        printf("Product found\n");
        break;
        case 6:printf("Exiting.....\n");
        break;
        default :printf("Enter valid option\n");
        break;
       }
    } while (choice!=6);

}
void push()
{
    char n[10];
    struct Stack *t=(struct Stack *)malloc(sizeof(struct Stack));
    if(t==NULL)
    printf("Stack full\n");
    else
    {
         printf("Enter product : ");
         scanf("%s",n);
         t->next=head;
         strcpy(t->name,n);
         head=t;
    }
}
void pop()
{
    char n[10];
    struct Stack *t;
    if(head==NULL)
    printf("Stack is empty\n");
    else
```

```
    {
        t=head;
        head=head->next;
        strcpy(n,t->name);
        printf("Product %s removed from stack\n",n);
        free(t);
    }

}
void display()
{
    struct Stack*t;
    t=head;
    while(t!=NULL)
    {
        printf("%s\n",t->name);
        t=t->next;
    }
}
void top()
{
    if(head==NULL)
    printf("Stack is empty\n");
    else
    {
        printf("Product : %s\n",head->name);
    }
}
int peek()
{
    char n[10];
    struct Stack *t;
    t=head;
    printf("Enter product : ");
    scanf("%s",n);
    if(head==NULL)
    printf("Stack is empty\n");
    else
    {
        while(t!=NULL)
```

```
        {
            if(strcmp(t->name,n)==0)
            return 1;
            t=t->next;
        }
    }
    return 0;
}
/*Develop a stack to manage items in a shopping cart using a linked list.
Use a switch-case menu with options:
1: Add an item to the cart (push)
2: Remove the last item (pop)
3: View all cart items
4: Peek at the last added item
5: Search for a specific item in the cart
6: Exit*/
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
struct Stack
{
    char name[10];
    struct Stack *next;
}*head=NULL;
void push();
void pop();
void display();
void top();
int peek();
void main()
{
    int choice,f;
    do
    {
       printf("1: Add an item to the cart (push)\n");
       printf("2: Remove the last item (pop)\n");
       printf("3: View all cart items\n");
       printf("4: Peek at the last added item\n");
       printf("5: Search for a specific item in the cart\n");
       printf("6: Exit\n");
```

```
       printf("Enter choice : ");
       scanf("%d",&choice);
       switch(choice)
       {
        case 1:push();
        break;
        case 2:pop();
        break;
        case 3:display();
        break;
        case 4:top();
        break;
        case 5:f=peek();
        if(f==0)
        printf("No item found\n");
        else
        printf("Item found\n");
        break;
        case 6:printf("Exiting.....\n");
        break;
        default :printf("Enter valid option\n");
        break;
       }
    } while (choice!=6);

}
void push()
{
    char n[10];
    struct Stack *t=(struct Stack *)malloc(sizeof(struct Stack));
    if(t==NULL)
    printf("Stack full\n");
    else
    {
         printf("Enter item : ");
         scanf("%s",n);
         t->next=head;
         strcpy(t->name,n);
         head=t;
    }
```

```
}
void pop()
{
    char n[10];
    struct Stack *t;
    if(head==NULL)
    printf("Stack is empty\n");
    else
    {
        t=head;
        head=head->next;
        strcpy(n,t->name);
        printf("Item %s removed from stack\n",n);
        free(t);
    }

}
void display()
{
    struct Stack*t;
    t=head;
    while(t!=NULL)
    {
        printf("%s\n",t->name);
        t=t->next;
    }
}
void top()
{
    if(head==NULL)
    printf("Stack is empty\n");
    else
    {
        printf("Item : %s\n",head->name);
    }
}
int peek()
{
    char n[10];
    struct Stack *t;
```

```
    t=head;
    printf("Enter item : ");
    scanf("%s",n);
    if(head==NULL)
    printf("Stack is empty\n");
    else
    {
        while(t!=NULL)
        {
            if(strcmp(t->name,n)==0)
            return 1;
            t=t->next;
        }
    }
    return 0;
}
/*Implement a stack to record payment history using a linked list. Include
a switch-case menu with options:
1: Add a new payment record (push)
2: Remove the last payment record (pop)
3: View all payment records
4: Peek at the latest payment record
5: Search for a specific payment record
6: Exit*/
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
struct Stack
{
    char id[10];
    struct Stack *next;
}*head=NULL;
void push();
void pop();
void display();
void top();
int peek();
void main()
{
    int choice,f;
```

```
    do
    {
       printf("1: Add a new payment record (push)\n");
       printf("2: Remove the last payment record (pop)\n");
       printf("3: View all payment records\n");
       printf("4: Peek at the latest payment record\n");
       printf("5: Search for a specific payment record\n");
       printf("6: Exit\n");
       printf("Enter choice : ");
       scanf("%d",&choice);
       switch(choice)
       {
        case 1:push();
        break;
        case 2:pop();
        break;
        case 3:display();
        break;
        case 4:top();
        break;
        case 5:f=peek();
        if(f==0)
        printf("No record found\n");
        else
        printf("Record found\n");
        break;
        case 6:printf("Exiting.....\n");
        break;
        default :printf("Enter valid option\n");
        break;
       }
    } while (choice!=6);

}
void push()
{
    char n[10];
    struct Stack *t=(struct Stack *)malloc(sizeof(struct Stack));
    if(t==NULL)
    printf("Stack full\n");
```

```
    else
    {
         printf("Enter transaction id : ");
         scanf("%s",n);
         t->next=head;
         strcpy(t->id,n);
         head=t;
    }
}
void pop()
{
    char n[10];
    struct Stack *t;
    if(head==NULL)
    printf("Stack is empty\n");
    else
    {
        t=head;
        head=head->next;
        strcpy(n,t->id);
        printf("Transaction %s removed from stack\n",n);
        free(t);
    }

}
void display()
{
    struct Stack*t;
    t=head;
    while(t!=NULL)
    {
        printf("%s\n",t->id);
        t=t->next;
    }
}
void top()
{
    if(head==NULL)
    printf("Stack is empty\n");
    else
```

```
    {
        printf("Transaction : %s\n",head->id);
    }
}
int peek()
{
    char n[10];
    struct Stack *t;
    t=head;
    printf("Enter transaction id  : ");
    scanf("%s",n);
    if(head==NULL)
    printf("Stack is empty\n");
    else
    {
        while(t!=NULL)
        {
            if(strcmp(t->id,n)==0)
            return 1;
            t=t->next;
        }
    }
    return 0;
}
```